AL-TR-1991-0085-Vol-1

DESIGN SPECIFICATIONS FOR THE ADVANCED INSTRUCTIONAL DESIGN ADVISOR (AIDA) (VOLUME 1 OF 2)

Albert E. Hickey

Mei Associates, Incorporated
1050 Waltham Street
Lexington, MA 02173

J. Michael Spector
Daniel J. Muraida

HUMAN RESOURCES DIRECTORATE
TECHNICAL TRAINING RESEARCH DIVISION
Brooks Air Force Base, TX 78235-5000

January 1992

Final Technical Report for Period August 1989 – July 1991

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5000

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

DANIEL J. MURAIDA
Project Scientist

HENDRICK W. RUCK, Technical Director
Technical Training Research Division

RODGER D. BALLENTINE, Colonel, USAF
Chief, Technical Training Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1992	Final – August 1989 – July 1991

4. TITLE AND SUBTITLE
Design Specifications for the Advanced Instructional Design Advisor (AIDA) (Volume 1 of 2)

5. FUNDING NUMBERS
C – F33615-88-C-0003
PE – 62205F
PR – 1121
TA – 10
WU – 43

6. AUTHOR(S)
Albert E. Hickey
J. Michael Spector
Daniel J. Muraida

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Mei Associates, Incorporated
1050 Waltham Street
Lexington, MA 02173

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
Armstrong Laboratory
Human Resources Directorate
Technical Training Research Division
Brooks Air Force Base, TX 78235-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AL-TR-1991-0085-Vol-1

11. SUPPLEMENTARY NOTES
Armstrong Laboratory Technical Monitor: Daniel J. Muraida, (512) 536-2981

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This is the final report for the second phase effort on the Advanced Instructional Design Advisor (AIDA) project. An experimental system called XAIDA is described. The proposed XAIDA is intended to assist subject-matter experts in the design and development of computer-based instructional materials. The functional requirements for an automated and intelligent advisor that would be appropriate for Air Force technical development are presented, along with a system specification.

14. SUBJECT TERMS
automated instruction
computer-based instruction
courseware design
instructional development
interactive technology

15. NUMBER OF PAGES
184

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

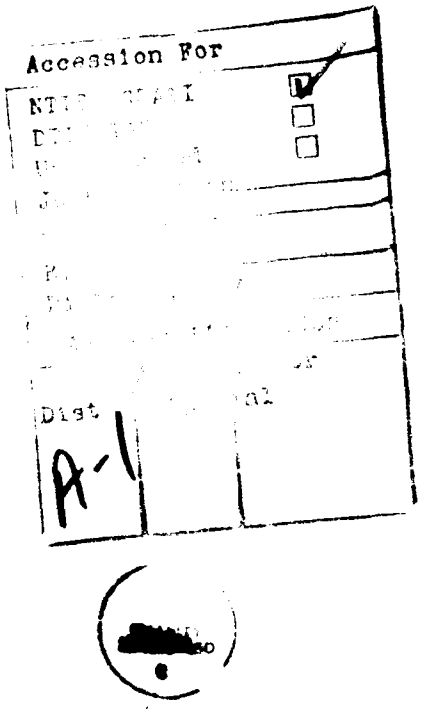


TABLE OF CONTENTS (CONT'D)

FIGURES

TABLE OF CONTENTS (CONT'D)

TABLES

APPENDIXES

PREFACE

This is the Final Report for Task Order No. 0013 of Contract No. F33615-88-C-0003. The report is the joint effort of the Armstrong Laboratory, Human Resources Directorate (ALHRD), the sponsor of the research, and the contractor, Mei Associates, Inc.

Section 1, describing the history of the AIDA concept within ALHRD and the Laboratory's long-range plan for AIDA, was written by ALHRD.

Sections 2 and 3, describing the purpose, detailed requirements, approach, and plan for Task 0013, were prepared by Mei Associates.

Section 4 presents an overview of the results obtained in Task 0013. It was prepared by Mei Associates.

The purpose of XAIDA is to assist subject-matter experts to design instruction for maintenance training. Section 5 is an analysis of the requirements for successful maintenance training derived from three papers prepared for Mei Associates by Dr. Henry M. Halff during Task 0013.

In Task 0006 it was agreed that AIDA would be based on the general theory of instructional design developed by Dr. M. David Merrill and his associates. Section 6 is a summary of that theory prepared by Dr. Merrill for Mei Associates during Task 0013.

In Section 7 the functional requirements for maintenance training described by Dr. Halff in Section 5, are restated in Dr. Merrill's more general instructional design theory, as presented in Section 6. This section, prepared by Mei Associates, comprises the functional baseline for XAIDA.

In Section 8 Mei Associates proposes a software architecture to implement the functional requirements for XAIDA. It is a summary of Appendix D, the Data Base Design Document.

Finally, Section 9 outlines additional research questions which were identified for both near-term and long-term resolution by ALHRD. This section was prepared by Mei Associates.

Mei Associates compiled the appendixes and edited the complete manuscript.

Dr. Dan Muraida was the contract monitor for ALHRD. Dr. Mike Spector conceptualized the AIDA project under the direction of Dr. Scott Newcomb, Branch Chief for the Training Technology Branch of the Training Systems Division of ALHRD. Dr. Albert E. Hickey was project manager for the contractor, Mei Associates.

SUMMARY

This is the final report for the second phase effort on the Advanced Instructional Design Advisor (AIDA). It consists of two volumes. The first contains the report; the second contains all of the appendixes. AIDA is an effort aimed at providing on-line and intelligent assistance to developers of computer-based instruction (CBI). Contained in this report is a summary of the project's methodology and progress. The results of the second phase are a set of functional requirements and a software system specification for an experimental system (XAIDA) which would satisfy those requirements. It is proposed that XAIDA be built around a second generation instructional theory involving the use of highly flexible transaction shells (Merrill, 1990). Transaction shells have built-in intelligence with regard to the pedagogy of particular types of subject matter. In addition, they are configurable and executable, which should result in a highly productive and effective courseware authoring environment. XAIDA will support the construction of scenarios and simulations so as to facilitate student acquisition of mental models considered crucial to instruction involving complex problem-solving tasks. It is recommended that the Air Force build and evaluate XAIDA.

DESIGN SPECIFICATIONS FOR THE ADVANCED INSTRUCTIONAL DESIGN ADVISOR (AIDA): VOLUME I

SECTION 1. BACKGROUND*

The purpose of this section is to provide background information. The discussion is divided into four parts: 1) a short history of the AIDA concept, 2) ALHRD/IDC's long range plan for AIDA, 3) the long range schedule, and 4) the benefits of the long range AIDA program.

1.1 History of the AIDA Concept

The Advanced Instructional Design Advisor (AIDA) was first described in Preliminary Design Considerations for an Advanced Instructional Design Advisor, Dr. Michael Spector's final report for his 1988 Summer Faculty Research Program grant at the Armstrong Laboratory Human Resources Directorate (ALHRD) [formerly Air Force Human Resources Laboratory (AFHRL)]. That research was conducted under the supervision of Dr. Scott Newcomb, Branch Chief for the Training Technology Branch of the Training Systems Division (ALHRD/IDC).

Newcomb assigned Spector the task of conceptualizing a next-generation courseware authoring system for the U.S. Air Force (USAF). Spector conducted an extensive literature search, reviewed several authoring systems used by the DoD including the Air Force's ISS and SOCRATES and the Navy's AIM and CBESS systems, and had many engaging discussions with Newcomb and other researchers about what the future held in store for courseware authoring.

The central problem in courseware authoring was identified as the difficulty and expense of designing effective instructional materials given the complexities of advanced hardware and software technologies and the variety of instructional settings. Existing systems did not address the issue of effective instructional design at the course level.

Spector proposed a computer-based tool to assist in the instructional design process. The tool would reduce course development time while assisting in the production of consistently effective instructional materials. AIDA would incorporate prescriptive advice about course authoring based on established theories of knowledge, learning, and instruction.

* This section was prepared by ALHRD/IDC.

AIDA would contain a variety of tools. Some would automate established processes. Others would assist or advise authors about designing effective instructional materials. The tools might be used as stand-alone, special purpose tools, or as integrated tools using a shared database of course and content information.

Since a variety of tools were envisioned, it was recommended that a modular approach be adopted and that a standard design philosophy be specified in order to provide for the graceful growth of the system. To incorporate the best instructional knowledge available, Spector recommended using a team of experts in the fields of epistemology, cognitive psychology, artificial intelligence, computer systems, and curriculum and instructional design. The team of experts would develop specifications for a standard design philosophy and a requirements specification for AIDA. A prototype AIDA based on a minimal functional subset would then be built and evaluated.

ALHRD/IDC decided to continue the exploratory development of AIDA under Work Unit 1121-10-43, Computer-Based Training (CBT) Software Development and Technical Support. The AIDA project is primarily a response to the Air Training Command (ATC) MPTN 89-14T, Research and Development of Computer-Based Instruction (CBI).

In a follow-on 1989 Research Initiation Program grant, Spector submitted a report entitled Refinement Considerations for an Advanced Instructional Design Advisor. In that document, the instructional design process was divided into three phases: 1) front-end analysis (FEA), 2) design, development, and delivery (DDD), and 3) rear-end analysis (REA). One finding was that the design phase of DDD, as well as FEA and REA, had received inadequate support in the form of research and development.

Spector also evaluated the potential role for artificial intelligence (AI) in the instructional design process. AI applications were divided into two large groups: artificial neural networks and expert systems. While neural networks hold great promise in the general area of pattern recognition, there does not appear to be an immediate application in the area of instructional design. As automated learning environments become more interactive and conversational, perhaps neural networks will have a significant role to play in the area of speech processing.

There does appear to be a significant role for expert systems in instructional design. Diagnostic expert systems have already been successfully incorporated into intelligent tutoring systems. Expert planning systems were envisioned to aid courseware designers in the development of consistently effective course materials for a variety of subject matter domains and knowledge types.

Task 13 is the second in a series of tasks (see Section 1.2 below) to refine, elaborate, and evaluate the AIDA concept. This report reflects progress on AIDA through 8 February 1991. The ALHRD/IDC AIDA Project Manager is Dr. Dan Muraida. The contractor for Task 13 is Mei Associates, Inc. of Lexington, Massachusetts. Mei Associates' Principal Investigator is Dr. Albert Hickey. Spector has continued working with ALHRD on the AIDA project under a University Resident Research Program grant.

1.2 The Long Range Plan for AIDA

This section represents ALHRD/IDC's current overarching plan for the AIDA project.

The purpose of the AIDA program is to provide reliable, effective, and efficient instructional design guidelines for USAF course designers. The approach to this goal contains four thrusts. Each is described below.

1.2.1 Status and Structure of Instructional Design Knowledge

The first thrust is an effort to assess what is currently known about instruction and instructional design. This assessment focuses on the major validated findings of instructional design research. The purpose is to derive critical instructional design information and initial estimates of the necessary detail for instructional design guidelines. This effort will identify areas of contradictory research findings and produce recommendations for an instructional design research and development agenda. It was conducted under a task order contract (Task 6).

1.2.2 Instructional Design Guidelines: Functional Requirements

The second thrust attempts to identify optimal methods of organizing and presenting validated instructional design information in the form of guidelines. Part of this effort builds on prior work which will have identified relationships among different areas of instructional design knowledge.

The focus is on functional requirements for an organized collection of guidelines which accurately represent the complexities of instructional design without exposing the user to those complexities. The functional requirements will be based on a model of instructional design which embraces the processes of design, development and delivery. A subsequent task will elaborate the functional requirements and develop a set of design specifications for an automated system of guidelines. This work was conducted under two task order contracts (Tasks 6 and 13).

The computer architecture and human factors issues involved in producing automated guidelines will be studied in a Small Business Innovation Research (SBIR) effort.

1.2.3 **Empirical Studies of Instructional Guidelines Delivery and Supporting Instructional Strategies**

In the third thrust, empirical research will be planned in parallel with the development of elaborated functional requirements and specifications. The elaborated functional requirements and specifications will encompass a complete automated guideline system, although they will be used to build an experimental testbed system which implements only certain basic functions of an automated guideline system.

All empirical research done prior to the development of the experimental testbed system will be conducted under the auspices of an in-house work unit. All development work devoted to building the testbed will be conducted under a Broad Agency Announcement (BAA) contracting vehicle. Empirical research using the testbed system to conduct studies on guideline characteristics or instructional strategies will be conducted under the auspices of the BAA.

a. Prior to Completion of the Experimental Testbed:

Guidelines research will focus on identifying critical variables in the automated instructional design process. Existing prototypes which automate a limited number of basic instructional design processes will be used for this purpose. The data and the conclusions of this stage of the guidelines research will provide useful information for design of the experimental testbed system and the subsequent research which it will support.

Research in the effectiveness of instructional strategies with computer-based interactive technologies will also employ existing instructional authoring tools. The first set of instructional strategy studies will address the lack of instructional principles for the use of audio reinforcement in a CBT setting. A second set of studies will examine Merrill's transaction theory and the usefulness of transaction shells in CBT. The results of the work accomplished prior to the completion of the experimental testbed system, will add to the knowledge base of instructional design principles required for a system of instructional guidelines.

b. After Completion of the Experimental Testbed:

At this point most of the guidelines research will be conducted using the testbed's capability to present different types of instructional design guidance and different methods of interacting with the user. Likewise, most of the instructional strategy research will use the capability of the testbed system to implement basic instructional functions (e.g., establishing linkages between instructional objectives and plausible instructional strategies).

The research and development conducted during this phase of the AIDA project will provide new information about the optimal nature and delivery characteristics of instructional design guidance for the non-expert instructional designer. Second, this research phase will begin to produce conclusions about the extent to which instructional strategies (new or old) can be extrapolated to a computer-based interactive setting.

1.2.4 **Empirical Test of an Instructional Design Guidance Methodology**

The fourth thrust of the AIDA research program ~~will~~ consist of design, development, and testing the complete experimental model. This will include tests of all functions of the automated guideline system, continued experimentation on optimal guideline characteristics, and continued research on instructional strategies in the automated delivery of instruction. This work will be conducted under a fully specified contract.

1.3 **The Long-Range Schedule**

1.3.1 **Milestones**

Complete work in first thrust area with the following products:

Review of current instructional design theory;
A cognitive model of the instructional design process;
Functional requirements for instructional design guidelines;
A research and development agenda for instructional design;
Recommendations for instructional design research.
.....3rd QTR90

Business Strategy: Task Order Contract

Contract award for functional and design specifications for a subsystem of instructional design guidelines ...3rd QTR90

Business Strategy: Task Order Contract

Completion of instructional design guidelines 1st QTR91

Contract award for (1) design, development and testing an experimental model of an automated guideline system and (2) initial instructional strategies and guidelines research 1st QTR91

Business Strategy: Broad Agency Announcement

Completion of experimental model tests and associated instructional strategies and guidelines studies...1st QTR94

Contract award for (1) Construction and test of the complete experimental model of automated guideline system and (2) continuation of instructional strategies and guidelines research 1st QTR94

Completion of experimental model tests and instructional guidelines and strategies research 1st QTR98

1.3.2 **List of Relevant Work Units**

Thrust:
1st: 1121-10-43, Task 6
2nd 1121-10-43, Task 6 and 13; 1121-10-70 (SBIR)
3rd 1121-10-66, In-House Work Unit; 1121-10-71 Direct BAA
4th 1121-10-67, Fully Specified Contract.

1.4 **Benefits of the Long-Range AIDA Program**

1.4.1 **Payoff**

The expected contributions of AIDA to Air Force training are:

- reduced training costs
- increased utility of CBT technologies
- increased instructor productivity
- reduced student time under instruction
- increased student comprehension and learning transfer
- establishment of instruction standards
- improved quality assurance.

1.4.2 **Opportunity**

Subject Matter Experts (SMEs) with little or no experience in instructional design or ISD will be able to use a theoretically oriented and empirically validated collection of instructional design tools to determine optimal instructional designs. As a result, novice instructional designers will be able to produce efficient and effective instruction in a variety of delivery modes.

SECTION 2. INTRODUCTION TO PHASE II (TASK 0013)*

2.1 Introduction

The purpose of the AIDA is to make the creation of training materials, particularly computer-based training (CBT), more cost effective through the application of advances in cognitive, instructional, and computer sciences. The goal in Phase II was to develop design specifications to support knowledge base development of AIDA and present recommendations for resolving instructional issues pertinent to knowledge base content. The design specifications will be the basis for building an experimental AIDA subsystem in Phase III.

2.2 Scope

The development of AIDA is proceeding in several tasks or phases. In Phase I, the objective was to design, develop, and document the concept and functional specifications for AIDA, as the first step in building an experimental AIDA subsystem in a subsequent task.

In Phase II, the objectives were to: (1) Develop a plan to resolve discrepancies or close gaps in the justifications for the use of an integrated instructional theory; (2) Identify implications of a (revised) integrated instructional theory for instructional research; (3) Describe the knowledge base sufficient to support the varieties of knowledge to be represented in the AIDA model; (4) Document the feasibility of continuing the development of the AIDA model.

2.3 Background

In Phase I of the AIDA project (Task 0006), (1) the AIDA concept was defined, (2) the functional characteristics were defined, and (3) related research issues were listed. The functional requirements of AIDA were related to underlying theories of knowledge, cognition, learning, instruction, and instructional design, and the way in which AIDA might be incorporated into the ISD procedure was described. Existing advanced authoring aids were evaluated.

2.4 Brief Review of Phase I Procedure (Task 0006)

To accomplish Phase I, Mei Associates assembled a team of seven consultants among the most creative psychologists in both theoretical and practical instructional design. Five of these seven psychologists functioned as design consultants and two as a review panel. In each of two five-month design cycles, the design consultants were charged with specific tasks. The tasks, in the form of concept papers, were then critiqued by the reviewers.

* This section was prepared by Mei Associates, Inc.

At the same time, the ALHRD organized a team of nine military advisors from the Army, Navy, and Air Force to comment on the utility of the functional characteristics specified by the design consultants. ALHRD also performed a needs analysis at the request of the consultants.

The 12-month design process was focused on five meetings: a kick-off meeting and four concept design meetings, held at ALHRD. Documentation was extensive, consisting of (1) the needs analysis, (2) fourteen concept papers, (3) four review papers, (4) the proceedings of the meetings, (5) progress reports prepared by ALHRD, and (6) the final report/functional specification prepared by Mei Associates.

2.5 Mei Associates Approach to Phase II (Task 0013)

2.5.1 Rationale

A principal product of Phase I, Cycle 1 was the AIDA concept, shown in Figure 2.1. It consists of six functions: five knowledge bases and the executive, as listed here:

1. Information about the:
   students
   instructional environment
   maintenance task
2. Subject-matter content
3. Executive
4. Instructional strategies
5. Instruction delivery
6. Evaluation

The output of AIDA Phase I also included statements of the facts, rules, heuristics, and procedures used in instructional design. These psychological statements were solicited from the experts -- the consultant psychologists -- in Phase I as functions, variables/attributes/characteristics within functions, and scales of values for the variables.

The further definition of AIDA in Phase II resulted in refinements of the functional specifications produced in Phase I. Another requirement was to integrate the psychological statements obtained in Phases I and II within a uniform systems framework. Since AIDA is a knowledge-based system, Mei Associates used the conventions of knowledge-based or expert systems, including production rules, frames, and an object-oriented language.

Transposing the psychological statements that make up the knowledge domain into conventions used in expert systems revealed gaps in the integrated theory of instruction design.

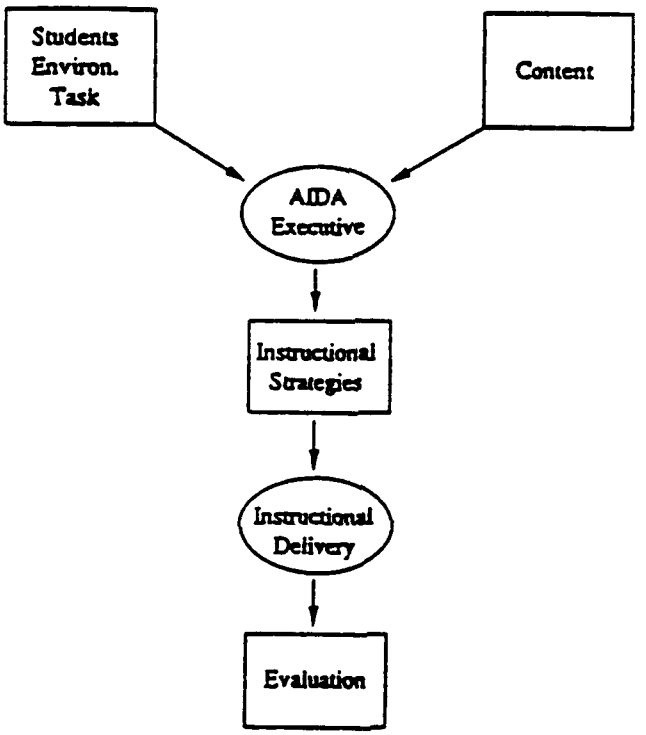


Figure 2.1 AIDA concept.

2.5.2 Procedure

There are two major components in an expert or knowledge-based system: (1) the knowledge content, and (2) the system architecture. To accomplish Phase II, Mei Associates assembled a team consisting of (1) experts in the knowledge domain of instructional design, and (2) a knowledge engineer (KE). The domain experts included the seven consultant psychologists employed in Phase I, plus an eighth consultant, Dr. Douglas M. Towne. Organization of the Phase II team is shown in Figure 2.2.

The result of knowledge acquisition is a specification of the knowledge contained in the expert system, in this case AIDA. Methods for the representation of the knowledge in the expert system must be developed. The expert knowledge is represented in frame- and object-oriented design. In Phase III, the representation scheme for the expert knowledge will be implemented in a prototype system for test and evaluation.

The basic characteristics of the knowledge representation scheme were defined during Cycle 1 of Phase II and presented in a technical interchange meeting attended by domain experts, KE, and representatives of ALHRD four months after task assignment. A schematic representation of the expert system proposed during the kick-off meeting is shown in Figure 2.3.

2.5.3 Knowledge Acquisition

Knowledge was gleaned from the domain experts during Phase II. These experts, including the consultant psychologists, pointed out the most useful and reliable information, and structured testable hypotheses where the necessary data were lacking. The domain experts also drew from refereed psychological journals the theoretical and/or empirical justification for the domain knowledge. Where facts, rules, heuristics, or procedures could not be found in the literature, or where the data reported in the literature were equivocal, the psychologists formulated testable hypotheses to resolve the issues empirically.

2.5.4 Expert System Architecture

Modeling of the AIDA system in Phase II was the result of interaction between the Mei Associates psychologist, representing the knowledge domain (i.e., instructional design), and the Mei Associates KE in an iterative process. Transposing the psychological facts, rules, heuristics, and procedures into more rigorous paradigms disclosed conflicts, missing links, etc. in the knowledge domain.

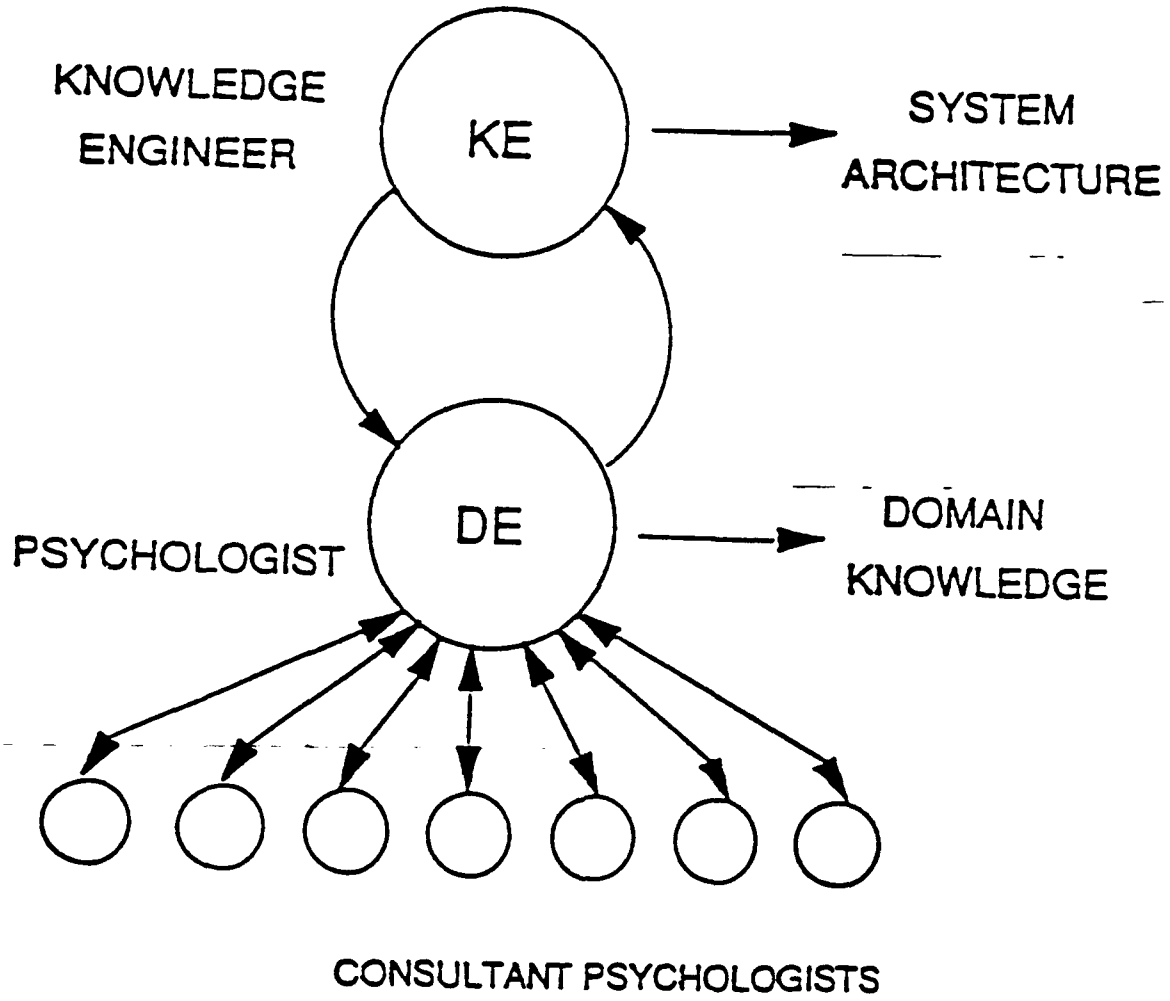
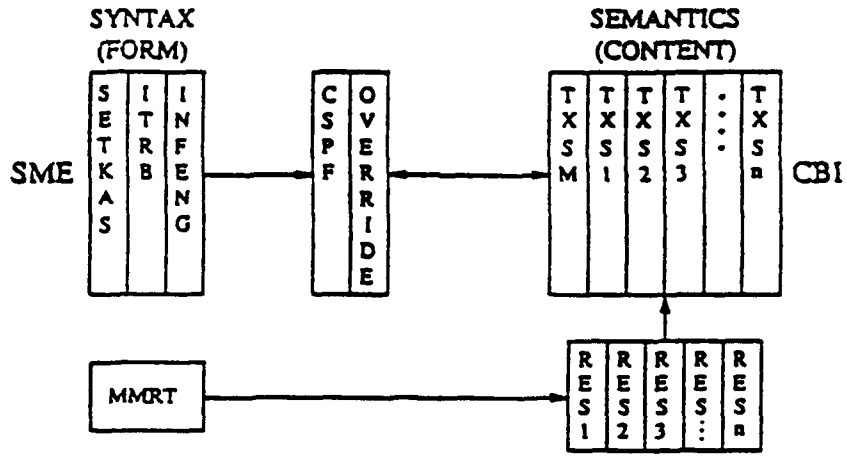


Figure 2.2 Organization of the Phase II team.



SME = Subject Matter Expert

SETKAS = Student, Environment, Task Knowledge Acquisition System

ITRB = Instructional Theory Rule Base

INFENG = Inference Engine

CSPF = Course-Specific Parameter File

TXSM = Transaction Shell Manager

TXS1 ... TXSn = Transaction Shells

MMRT = Multi-media Resource Toolkit (Paint, Draw, Scanner, Audio)

RES1 ... RESn = Resource Files

CBI = Computer Based Instruction

Not Shown: Help System
Audit Trail

Figure 2.3 Schematic of AIDA Expert System.

2.5.5 **Demonstration of the AIDA Model**

A paper-based demonstration of the AIDA instructional design model was performed by using the model to develop a minimal application, a course in troubleshooting aviation equipment. This exercise employed a subset of the facts, rules, heuristics, and procedures in each of the knowledge bases and the executive: Information, Subject matter, Executive, Strategies, Delivery, and Evaluation (see Figure 2.1). The domain experts, both consultant psychologists and military advisors, were asked to evaluate the model in TIM-2.

Mei Associates, in consultation with ALHRD, selected the application, the T-38 engine start scenario, after considering the F-16, and WSC-3, a shipboard communication system.

The team of 12 military advisors assembled by ALHRD for Phase I continued to play an important role in Phase II; they received copies of documents and attended technical interchange meetings.

SECTION 3. METHODOLOGY FOR PHASE II*

3.1 The Consultants

To accomplish Phase II, Mei Associates carried forward as consultants the seven psychologists retained in Phase I, and added an eighth consultant. The eight consultants were chosen from among academic researchers with a reputation and demonstrated research in fields related to instruction design. They are:

1.	Dr. Robert M. Gagne	Florida State University
2.	Dr. Henry M. Halff	Halff Resources, Inc.
3.	Dr. M. David Merrill	Utah State University
4.	Dr. Martha C. Polson	University of Colorado
5.	Dr. Robert D. Tennyson	University of Minnesota
6.	Dr. Harold F. O'Neil	University of Southern California
7.	Dr. Charles Reigeluth	Indiana University
8.	Dr. Douglas M. Towne	University of Southern California

3.2 The Military Advisors

At the same time, the Armstrong Laboratory Human Resources Directorate (ALHRD) carried forward from Phase I a board of Military Advisors. The 14 military advisors, representing the three services, are:

1.	Dr. Dee Andrews	ALHRD/OTT
2.	LCOL Jerry Barucky	USAFRS/RSCD
3.	LCOL Mike Bush	USAFA/DFF
4.	LCOL Larry Clemons	ATC/XPCRI
5.	CAPT James Coward	HQ ATC/XPCRI
6.	Mr. Brian Dallman	3400 TCHTW/TTOZLCOL
7.	LCOL Mike Dickinson	HQ HSD/YA
8.	Dr. John A. Ellis	NPRDC
9.	Dr. Mary Marlino	USAFA/DFT
10.	Dan Meigs (Retired)	3302d TCHTS/CC
11.	MAJ Robert Mongillo	ATC/XPCRR
12.	Rich Ranker	AWC/DFP
13.	MAJ Karen Reid	ATC/TTIP
14.	Dr. Robert Seidel	Army Research Inst.

3.3 Project Management

The Project Manager for the contractor, Mei Associates, was Dr. Albert E. Hickey. The Project Monitor for ALHRD was Dr. Daniel Muraida.

* This section was prepared by Mei Associates, Inc.

3.4 **The Procedure**

As described in detail in Sections 1 and 2, the objective of Phase II (Task 0013) was to design, develop, and document the system specifications for AIDA. The nine-month task was organized around three two-day meetings: A Kick-off Meeting and two Technical Interchange Meetings (TIM-1 and TIM-2). There was also a three-day meeting of a four-person working group interpolated between TIM-1 and TIM-2 and a concluding three-day meeting between ALHRD and Mei Associates, Inc. to review the data items to be delivered by the contractor. The schedule is shown here.

TABLE 3.1 PHASE II SCHEDULE

1.	Kick-off Meeting	26 Apr 90
2.	TIM-1	1-2 Aug 90
3.	Working Group Meeting	3-7 Sep 90
4.	TIM-2	7-8 Nov 90
5.	Final Technical Review	15-17 Jan 91

3.5 **Technical Reports**

As Phase II progressed and the functional specifications became more detailed, technical interchange among the consultants was enhanced by the exchange of working papers, orchestrated by the Project Manager. A diagram of information flow during the last three months of Phase II is shown in Figure 3.1. The key to Figure 3.1 is given below. The documents are also listed in Appendix A: List of Documents Produced.

Merrill

G'L TH./ID (ID2)	General Theory of Instructional Design
IXT/MT	ID Theory for Maintenance Training
FBL/XAIDA	Functional Baseline/XAIDA

Halff

AMT	Automating Maintenance Training
TTP	Teaching Troubleshooting Procedures
LTR	Letter Report

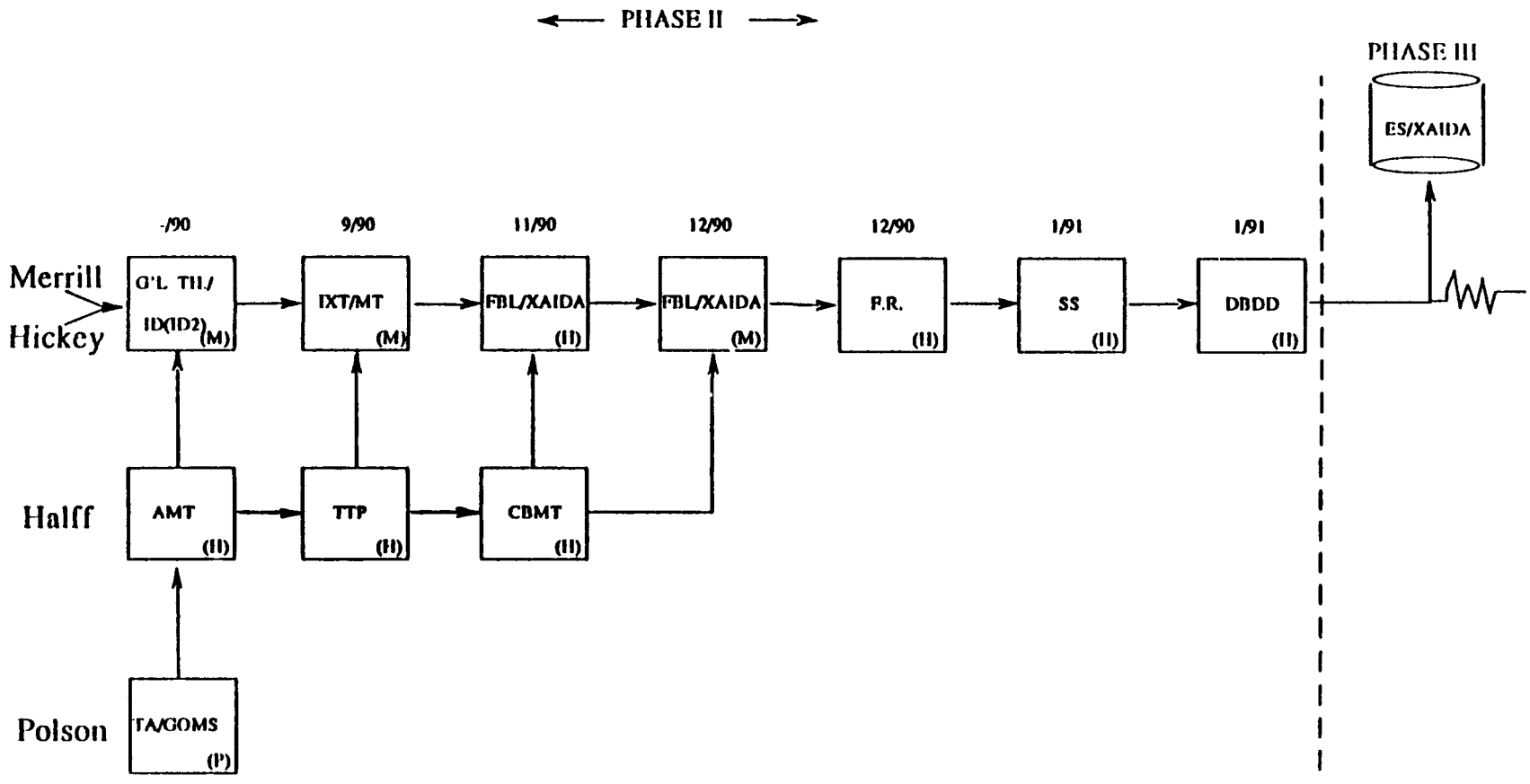


Figure 3.1 Documentation flow diagram.

Polson

TA/GOMS	Cognitive Task Analysis (with GOMS)

Hickey

FBL/XAIDA	Functional Baseline/XAIDA
F.R.	Final Report
SSS	System/Segment Specification

Mei Associates

DBDD	Database Design Document
ES/XAIDA	Expert System/XAIDA

SECTION 4. OVERVIEW OF TASK RESULTS

4.1 Introduction

In Cycle 1 of AIDA Phase I, the design consultants were asked to "provide a concept of how AIDA would support the design of instruction...and identify the principles of learning and instruction that apply to the design of the two AIDAs and to their instructional products." This resulted in the AIDA concept diagram. In Cycle 2, the consultants were asked to respond to a set of questions about the AIDA concept. The eventual result was a list of 21 guidelines for XAIDA, listed below:

Key System Attributes

1.	Targeted Users:	intermediate
2.	Type of Assistance:	templates, shells and explanations
3.	Subject Domain(s):	avionics maintenance
4.	ID Theory:	ID-2
5.	Starting Input:	learning capabilities
6.	Halff's Three Levels	facilitator (using ID-2)
7.	Tennyson's Three Levels	shells and some sophisticated shells
8.	Control of ID Process:	primarily user control
9.	Priority Components:	procedure transaction shells
10.	System Organization:	Merrill
11.	ISD Scope:	DDD not bound by AF model
12.	Use of AI:	intelligent interface to transaction shells
13.	Media Support:	text, graphics, audio, video
14.	Use Interface:	pull-down menus with buttons
15.	Data Collection:	built-in
16.	Databases:	instructional rulesets, and examples

17.	On-Line Support:	help, limited authoring mgt. and storyboard
18.	Learning Theory:	existing theories
19.	Instructional Setting:	computer-based
20.	Organizing Strategy:	transactions
21.	Knowledge Acquisition:	built-in

During Phase I, then, the design of the AIDA became more focused. This process of convergence continued in Phase II, and was accelerated in the last few months. Figure 3.1 is a flow diagram showing, paper by paper, the exchange of information leading to the three CDRL items: the System Specification (SS), the Data Base Design Document (DBDD), and the Final Report (FR). Sections 5, 6, and 7 which follow, contain abstracts of selected papers developed by Mei Associates and its consultants during this period of convergence.

The Data Item Description (DID) for the System/Segment Specification (SS) describes the SS as the "Functional Baseline" (FBL) for the system, in this case AIDA. Section 5 is an abstract of Halff's three papers describing the instructional design task for maintenance training. Halff's description of maintenance training for the T-38 engine start system is the functional baseline most specific to the demonstration instructional design (ID) task. Certain concepts used in maintenance training, such as the fault tree, are discussed in depth, i.e., (1) how the SME/ID builds it, (2) how the S interacts with it, (3) how it fits into a lesson or course, etc. (ALHRD intends to publish each of these papers as a Technical Report. See References.)

Guideline #4 in the above list declares that, to gain generality, AIDA should be built around Merrill's second generation theory of instructional design (ID-2). Section 6 is a description of ID-2 written after Merrill had studied Halff's description of maintenance training (Section 5.2.6). Merrill's paper accommodates several concepts introduced by Halff.

It should be possible to restate Halff's description of maintenance functions in the language of Merrill's more general theory of ID-2. Section 7 is a function-by-function abstract of Halff's paper, with each function accompanied by a transcript in Merrill's ID-2 language. Some Halff functions, such as "Describe the system," slip easily into the ID-2 scheme; other Halff functions, like "Develop a hierarchical fault tree," do not.

Halff, asked to comment on the step-by-step transcript of his functional baseline and to visualize in more detail the screen-by-screen interaction between XAIDA and the SME/ID, responded by describing an authoring system in terms of the authoring

functions provided to the SME/ID and the instructional products provided by the system (see Section 5.4).

Halff agreed that the instructional products should be built up from transactions as defined in ID-2.

Merrill, in turn, was asked to look at Halff's functional baseline and describe how Halff's (H) functions could be expressed in ID-2. He responded with the three-column table presented in Section 5.5.

4.2 ISD and AIDA

It was agreed in Guideline #11 that in XAIDA instructional design, development, and delivery (DDD) would not be bound by the USAF model for Instructional System Design (ISD). In fact, however, the functions to be implemented in XAIDA and later in AIDA coincide with several steps in the ISD process, particularly the ISD steps to:

I.1	Analyze job (task)
I.2	Select tasks/functions
II.4	Determine sequence and structure
III.1	Specify learning events/activities
III.3	Select materials
III.4	Develop instruction
III.5	Validate instruction
IV.2	Conduct instruction

The output of XAIDA and AIDA represents microinstruction when compared with the ISD process. It is expected that a closer correlation between the two systems will occur later in the AIDA life cycle.

4.3 Task Analysis

One conclusion reached in the final planning session in Phase II was that "some kind of task analysis will be provided within XAIDA." The kind of task analysis to be provided in XAIDA is discussed in Section 5.5.

5. INSTRUCTIONAL DESIGN FOR MAINTENANCE TRAINING

5.1 Introduction

Henry Halff contributed three valuable papers to Phase II, all dealing with maintenance training. In the three papers, Halff takes maintenance training from (1) a consideration of general principles drawn from cognitive psychology, through (2) the application of the principles to a particular case, the T-38 maintenance scenario, to (3) a brief description of a computer-based authoring system based on the concepts developed in the earlier papers. This section is an abstract of Halff's three papers, taken in series. The reader is urged to read the original papers for important information necessarily omitted from this abstract.

5.2 Automating Maintenance Training (AMT)

Halff's first paper (Halff, 1990) is divided into two sections. In the first section, Halff describes six tasks that constitute effective maintenance and the mental structures that support proficiency in these tasks. Then he describes a curriculum that supports development of the mental structures required for proficiency.

The second section of the paper is concerned with automation of the training process and the development of materials. Halff suggests a design for automated, interactive maintenance training based directly on the results of the first section, and points out aspects of this design that can be automatically implemented by computers.

5.2.1 Instructional Design for Maintenance Training

5.2.1.1 Maintenance Tasks

Halff lists six tasks that a maintainer must master in order to effectively maintain a piece of equipment. The tasks are equipment oriented; they describe what must be done to the equipment. They do not say how the tasks are to be performed by the maintainer.

(1) Operation

In most maintenance contexts the maintainer must be able to operate, to some degree, the equipment being maintained. Operational skills are used to verify the status of the equipment, to prepare the equipment for maintenance, and to interpret reports from operators.

(2) Calibration and Adjustment

Many devices must be configured for particular operating environments, calibrated, and adjusted on occasion. Maintenance personnel are routinely called on to effect such adjustments.

These adjustments are often a part of preventive maintenance, and they often constitute repairs.

(3) Testing

Equipment testing is a critical part of maintenance. Maintainers must be able to test an equipment's operational status. They must also be able to conduct particular diagnostic tests during the course of troubleshooting. These tests often require the use of general-purpose and specialized test equipment, and this test equipment must itself be properly calibrated and operated.

(4) Access and Disassembly

In the course of repair, testing, and calibration, maintainers must gain access to particular components for observation and manipulation. The procedures used to gain access can be straightforward in some cases. In others, special procedures are required to ensure that gaining access to one part of the equipment will not damage other parts. These procedures are normally specified by the manufacturer of the device.

(5) Repair

By repair, Halff means the operations needed to restore the operation of a device to specifications once a fault has been isolated. Repairs include replacement of faulted components, cleaning, adjusting, patching, and a host of other operations.

(6) Troubleshooting

Perhaps the most challenging maintenance operation from a training viewpoint, troubleshooting is the process of identifying the physical cause (fault) of an existing or potential malfunction of the equipment's operational capabilities. For the most part, troubleshooting takes place after a malfunction occurs, but troubleshooting also comes into play when a test - for example, during preventive maintenance - reveals a potential fault.

5.2.1.2. Cognitive Components of Maintenance Skills

Halff then describes three mental structures that support the six maintenance tasks described above: i.e., (1) a mental model of the equipment, (2) the execution of fixed procedures, and (3) fault isolation skills.

(1) Mental Models of Equipment

Mental models, in the context of maintenance, are the cognitive structures used to reason about the equipment being maintained. They take account of the device's structure, function and physical manifestation.

Structure

Structural knowledge can be cast in terms of the equipment's components and its topology. The device is represented as a directed graph with individual components at the nodes. Each component is represented by a device model used to derive the component's outputs as a function of inputs. Components can operate in any of a number of modes, including fault modes, so that differential predictions can be derived for faulted and not faulted cases.

Reasoning within this model takes the form of propagating changes from component to component. Informed of an input or change in input to one component, the model derives the consequences for that component's outputs and then propagates the result to the components connected to the original component's outputs. The general term for this interpretation process is qualitative reasoning, as opposed to quantitative reasoning which derives predictions from the joint application of mathematical constraints (e.g., Kirchoff's law).

Function

Functional knowledge of a device (and its subsystems and components) is less well understood. Intuition argues that functional knowledge forms the cornerstone of structural knowledge. Functional knowledge can explain or rationalize the structure of a device and thus serve, at least, as a mnemonic for device structure. (For the importance of mnemonics in cognitive skills, see Chase and Ericsson (1982).) In addition, as was mentioned above, functional knowledge may be used to resolve impasses in qualitative reasoning. At the least, functional knowledge guides the technician's evaluation of the device's operational status.

Research on mental models of device functionality is sparse. One of the most interesting analyses can be found in Kieras (1988).

Imagery

Mental models are of little use unless they can be correlated in some fashion with the actual equipment. There is considerable evidence (Kosslyn, 1980) that imagery plays a significant role in making sense of the outside world. Indeed, some assumption about

imagery underlies any depiction of the actual equipment - pictures, high fidelity simulators, the equipment itself - in training and documentation.

Imagery also plays an important role in providing internal cognitive support for the mental model. Graphics are pervasive in maintenance training and documentation, providing concrete imaginal representations of the abstract notions that make up the working part of the mental model.

The Importance of Mental Models

Kieras (1988) and Kieras and Boviar (1984) point to the importance of mental models in device operation. Some of their arguments apply also to maintenance. Looking at the list of six maintenance tasks, we can see the pervasive support provided by mental models.

(1) Mental models provide constraints that enable effective operation of the equipment with less than perfect memory of the operating procedures themselves.

(2) Calibration and adjustment are, by definition, goal-oriented procedures that are supported by a knowledge of the structural relations between controls and indicators.

(3) While some aspects of testing relate directly to the equipment's functionality, most tests are conducted to establish the condition of particular components or subsystems. In these cases mental models provide coherence to the formulation and interpretation of tests.

(4) The conceptual structure of equipment is at least roughly reflected in its physical structure. Single circuit boards, for example, often implement single modules. Thus, a mental model of the device offers considerable advantage to the technician who must disassemble that device to gain access to a particular subsystem, module or component.

(5) Whenever technicians face some choices in making repairs, their mental device models can be used to guide their decision. E.g., a basic knowledge of how conductors function makes it easier to determine how to repair an open circuit.

(6) Troubleshooting often amounts to testing hypotheses about the location of a fault. Mental models support this hypothesis-testing procedure by providing predictions from various hypotheses.

(2) Procedures

Looking at the list of six maintenance tasks, one is impressed by how often the maintainer is required to execute simple procedures.

Definition of a Procedure

Halff's definition of a procedure has three parts (which happen to correspond to the three aspects of mental models).

(1) A procedure's control structure defines the sequence of steps to be taken when executing the procedure and can be represented a number of ways, including augmented transition networks, and-or graphs, production systems, and others. All have the capability to represent both branching (decisions) and sequential constraints (stepwise progression).

(2) A procedure's function defines its teleology or goal. This term, like that of device function is theoretically dispensable (and, indeed, may never become apparent to some learners); however, most would argue that an understanding of a procedure's function is an indispensable part of its definition. At the least, functional knowledge supports the use of the procedure in problem solving activities.

(3) Procedures also interact with the outside world. In human terms, therefore, the perceptual-motor (input-output) concomitants of a procedure are an important part of its definition. When a procedure calls for, say, setting switches or reading dials, the technician must know how to accomplish these actions and make these observations.

Procedure Implementation

Effective execution of any procedure depends on getting procedural knowledge into the head of the technician, but how this knowledge gets in, when it gets in, and in what form are open questions dependent on the nature of the maintenance enterprise.

In some cases procedures must be fully represented in procedural form in the head of the maintainer. To take an extreme case, as soon as the fire light goes on in the cockpit of an aircraft, the pilot becomes a maintainer whose initial procedure for dealing with the situation must be highly automated.

In other cases, procedures can be interpreted, usually from descriptions found in job aids and other documentation. Both preventive maintenance and repairs in steam power plants are heavily guided by these sorts of procedures.

In still other cases, procedures are needed that are not available through training or documentation, but are invented by technicians to accomplish particular aims. For example, combat has been known to inflict widespread damage to weapon systems and their subsystems. The task of dealing with such widespread damage is not covered in detail in either training or documentation. Nonetheless competent technicians can often deal with these situations by redesigning the system on the basis of a mental model.

The invention of procedures to meet certain goals is a special case of problem solving, and the situations that require this sort of problem-solving in the maintenance arena are of two types: complex malfunctions and difficult repairs.

Complex malfunctions include such cases as multiple faults, intermittent malfunctions, faults in systems too complex for standard troubleshooting practices (e.g., feedback systems), unreliable test equipment, and parts inventories contaminated with faulty components. These problems have many of the characteristics of decision making under uncertainty.

Difficult repairs can be attempted when large portions of a system are damaged, when parts needed for particular repairs are not available, when the equipment or parts thereof are not accessible, or when other circumstances prevent normal repairs. Solutions to these problems often involve jury rigs in which some part of the system is redesigned and rebuilt to restore functionality.

(3) Fault Isolation

Troubleshooting is a problem in a space where devices are viewed as networks of components. One or more component is in a fault mode, and the troubleshooting problem is to devise a sequence of actions that isolate and repair the faulted components. Actions in the solution sequence include:

(1) Observe the outputs of some components.

(2) Observe the states of some components (e.g., LEDs).

(3) Manipulate the states of some components (e.g., switches).

(4) Replace certain components.

Costs, in terms of time and money, can be assessed for each of these actions.

Troubleshooting skills are a mix of "strong" knowledge-based methods and "weak" context-independent skills. Knowledge-based methods are associations between familiar patterns of observations and troubleshooting actions. Context free methods require analysis of the topology of the device to determine which

troubleshooting actions discriminate among a set of plausible or hypothetical faults.

In addition to these two selection principles, troubleshooters are also guided by the information-theoretic value of potential observations, and will choose those that provide the greatest reduction in uncertainty (Towne, Johnson, & Corwin, 1983).

Formal competence models of the troubleshooting process are available (Hunt and Rouse, 1984; Towne, Johnson, and Corwin, 1983) and are eminently suitable for use in training (Towne and Munro, 1988; Towne, Munro, Pizzini, Coller, and Wogulis, 1990). What is not available at this time are valid performance models of troubleshooting. For example, it is obvious that skilled troubleshooters do not compute the information associated with each potential observation in order to choose among them. Rather, they probably use heuristics based on a structured view of the device topology. Sorely needed is a theory of how technicians arrive at this structured view, and how they interpret the view in the course of choosing observations.

5.2.2 **Instruction of Maintenance Skills**

In this section Halff describes a curriculum that corresponds directly to the structure of the instructional objectives.

5.2.2.1. Orientation to the Equipment

Halff first describes the instruction that is needed to convey a mental model of the equipment being maintained. The main components of such a model are (1) the device's function, including its operation, and (2) the device's structure, including its topology and component behavior.

Device Function and Operation

Device function can be conveyed to students in a number of ways.

(1) The device can be identified as one of a larger class of devices whose functions are known to the student. What distinguishes the device from others of its class should also be made known to the student.

(2) The device, or a model thereof, can be shown in operation, preferably under student control, and preferably in such a way as to cover all of the major operating states of the machine.

Device Structure: Topology and Component Behavior

As with device function, any aspect of a device's structure can be identified as a member of a larger class with the same structural characteristics. Of more interest are mechanisms for directly conveying device structure. Traditional methods rely on

paper documentation and consist, usually, of a block diagram, a narrative description, and data on the specifications of each component. More recently, computer simulations such as STEAMER (Hollan, Hutchins, and Weitzman, 1984), have offered a number of benefits not to be found in conventional documentation. Among these are:

- o support for practice of causal reasoning,

  Students can be given partial information about the state of the device and asked to predict some aspect of device state not evident in the display. Both the sequence of exercises of this sort and the structure of feedback can be based on the structure of the device itself.

- o exhibition of complex component functions,

  Students can be shown (interactively) how a component or subsystem reacts to different combinations of inputs and changes in inputs. For example, students can be given some of the inputs to a component and asked how the others might be set to put the component in a particular state.

- o exhibition of component, subsystem, and device behavior under any normal or faulted operating states,

  Simulations of the sort described can be designed to reflect the conceptual structure of the device. Thus, for example, components that are structurally related can be shown in the same display even though they are physically separate in the equipment itself.

- o association of the mental model's components with their imaginal manifestations,

  Video and other devices can be exhibited in connection with their conceptual counterparts in a simulation.

- o presentation of mnemonics and other mechanisms for cognitive support of learning.

  Symbolic depictions of device components can be designed to reflect their state or function. The traditional use of icons in electronic diagrams is an obvious example of this mechanism.

5.2.2.2 Procedures

Earlier, Halff described three ways the maintainer can implement procedures:

(1) directly from the technician's procedural knowledge,

(2) from interpretation of a job aid, or

(3) through composition as the result of problem-solving (troubleshooting) activity.

To implement a procedure, attention must be paid to the control structure of the procedure, its function and its perceptual-motor aspects.

Procedural Knowledge

Certain well-known prescriptions apply to the teaching of procedures (in the sense of helping the student achieve unaided execution of the procedure).

- Depict the control structure of the procedure along with its function, and the constraints that the function imposes on the control structure.
- Make intermediate results available to students during training.
- Provide practice in the procedure under conditions that preserve a consistent mapping of stimulus to response.
- Provide examples that span the space of choices that must be made in executing the procedure.
- Provide enough practice with real equipment or high-fidelity simulators to ensure mastery of the sensory-motor aspects of the target procedures.

If a procedure has more than one choice point, make sure that the student is able to make each choice in isolation of the others.

Interpretation Skills

Training technicians to interpret written instructions is (or should be) more a matter of designing the instructions than training the students. The task of interpreting instructions is partly one of inducing the procedure from examples and partly one of deducing the procedure from written descriptions.

- Devise a procedure for interpreting the set of job aids used on the job and teach this procedure using the principles suggested above for Procedural Knowledge. The procedure should function to find the right job aid for the occasion and properly interpret the aid.

Problem-Solving Skills

Problem solving is almost always taught on the job, where most of the opportunities for training of this sort occur. Acquisition of problem solving skills can be promoted in school in several ways.

- Since effective problem solving depends crucially on a technician's facility with the mental model of the system being maintained, extensive practice with this model should enhance the ability to create effective procedures in unanticipated situations.

- Case study, at least by default, is the preferred method for promoting problem-solving expertise. In many situations, it should not be difficult to import this method from the job site to the schoolroom by recording and packaging selected cases.

- Certain general problem-solving techniques (e.g., decision theory, heuristic reasoning) seem to describe effective solutions so well that they bear considerable promise in instruction. Configuring these methods to difficult maintenance problems might yield substantial benefit.

5.2.2.3 Troubleshooting

Troubleshooting is generally taught through a series of troubleshooting exercises, and practice will no doubt be the backbone of the most effective methods of troubleshooting training. This said, what needs to be specified are the sequence of problems to be used in troubleshooting practice and the practice environment.

Problem Selection in Troubleshooting Practice

The aim of troubleshooting practice is to build both knowledge-based and context-independent skills. This goal suggests that certain problems, which exercise neither type of skill, should be excluded from consideration in practice sets, and that other problems should be included. Earlier, Halff (1989) listed a set of simplifying restrictions that exclude non-instructive problems:

In typical training situations, certain simplifying assumptions govern the behavior of the equipment.

- Every malfunction is the result of a single faulted component, although in real equipment multiple faults often occur.

- Faults can be characterized as a change in the state or possible states of a component, not in the topology of the equipment, although in real equipment faults can change the nature of the connections among components.

- Neither testing nor replacing a component will fault another component, although in real equipment a faulted component can protect another component from damage.

- Finally, we assume that there are no faulty replacements, even though real world technicians will on occasion return a faulted component to inventory.

In addition to these exclusion criteria, certain types of faults should always be included in troubleshooting practice.

- Mission critical malfunctions should be included so that students learn the pattern of observations associated with these faults.

- Principles of discrimination learning dictate that close relatives of mission critical malfunctions should be included in practice sets. These close relatives are those whose patterns closely match those of the corresponding mission-critical faults.

- To exercise context-free troubleshooting skills, students should be given a set of structure spanning malfunctions, i.e., faults should be chosen that allow for application of all context-independent strategies.

The Practice Environment

The practice environment is as critical to successful training as is the selection of problems. Many general principles of procedure learning also apply to troubleshooting practice.

- Density of practice is important. Simulators should be used and designed to engage students in as many practice problems as possible.

- Hidden cognitive operations should be made evident to the student. Thus, students should be forced to track, actively or passively, the results of each troubleshooting action. Typically this means they should identify which faults are eliminated by each action.

- Strategic information should be made evident to the student. This can be accomplished though the advice and critiques of a human or machine tutor.

5.2.2.4 Curricular Issues

The suggestions made above for meeting general instructional objectives present a significant challenge to curriculum design. The objectives themselves are interrelated and so it seems that the curriculum should reflect this interrelatedness. Any of the instructional paradigms that address the objectives can be configured in many different ways. Different components can be chosen for exercises, different degrees of support can be provided. The division of effort between student(s) and instructor can vary. It may, therefore, be of some use to recall a few general principles that have guided curriculum development.

Lesson Structure

The course should be divided into discernable lessons and/or other recognizable units. The goals of each lesson and the methods used to achieve those goals should be made clear to students.

Teach Prerequisites First

The subskills or prerequisites of a skill should be addressed in training before the skill itself. Thus, for example, since troubleshooting involves certain inferences to be made from a mental model of the device, exercises addressing these inferences should be provided prior to troubleshooting training.

Whole- and Part-Task Training

There are two approaches to training tasks that involve distinct subtasks. Part-task training calls for practice of the subtasks in isolation. The whole-task approach allows students to practice the entire task using external cognitive support for subtasks that have not yet been mastered. Maintenance training can, and should employ both methods. For example, whole-task training in diagnostic maintenance can focus on replacement procedures by allowing an expert to walk the student through the trouble-shooting stage of a repair. Part-task training in troubleshooting itself may be of some benefit because eliminating non-essential tasks such as equipment disassembly can increase the density of practice.

Completeness

When procedures or problem-solving skills involve branching, exercises should be provided that address all significant variation in input to these procedures. Thus, students should be given troubleshooting exercises that introduce them to all significant topological patterns in the structure of the equipment.

Fading

Any exercise can be configured with a variety of cognitive supports. Among these are techniques that exhibit the correct moves to the student, those that provide structural aids such as qualitative simulation, providing intermediate steps such as subgoals, and many others. Students should begin working in heavily supported environments and should graduate to successively less heavily supported environments until they are practicing in an environment close or identical to the actual task environment.

Review

Old material should be reviewed when new material is introduced. Review functions not only to space practice but also to show students how to discriminate the situations appropriate for the application of old and new skills.

The Trials Effect

Skill increases with the amount of practice.

5.2.3 **Computer-Based Maintenance Training**

5.2.3.1 Automating the Training Process

Interactive maintenance training (e.g., Towne, 1986) has generally been restricted to high-fidelity simulators. That is, simulators that physically resemble the equipment being maintained. This physical resemblance is implemented either with three-dimensional mockups, with film, and more recently, with video. Instruction usually consists of pure troubleshooting exercises, other aspects of maintenance training being covered by more traditional media.

Recent developments in instructional technology and the design philosophy described above suggest a different approach to interactive maintenance training systems. This new approach would differ from existing methods in several respects.

- Because of the primacy of mental models in effective maintenance, qualitative simulation with an explicit representation of a mental model would play a primary role in maintenance training.
- The scope of interactive instruction could be expanded to include instruction in reasoning from a mental model and procedure training (in addition to troubleshooting training).
- Functions such as critiques and coaching now often removed from the practice context would be incorporated into that context.
- Depictions of actual equipment (using video, sound, and other means) would be used for the explicit purpose of associating elements of the mental model with their real-world counterparts.

The remainder of this section contains some suggestions for implementing this approach.

(1) The Infrastructure of Interactive Maintenance Training

Halff strongly recommends the development of an infrastructure representing the knowledge to be acquired during training. Instructional methods for conveying this knowledge employ the infrastructure as a source of instructional material. Halff defines the infrastructure's main components in this section and makes some suggestions for its instructional use in the next section.

Qualitative Simulation

By qualitative simulation, Halff means a presentation of a device's function and structure in terms of the kind of mental model described earlier. This presentation could be implemented using a program such as the Intelligent Maintenance Training Simulator (IMTS) (Towne and Munro, 1988; Towne et al., 1990) where the mechanics of the mental model form the basis of the simulation program itself, or a computational approach such as the mathematical model of a steam plant used in STEAMER (Hollan, Hutchins, and Weitzman, 1984). What is important for our purposes is that the simulation appear to the student in the form of a mental model. This means that

- the topology of the device is reflected in the display by showing the connections among components;
- the conceptual structure of the device should be reflected in the simulation by appropriate grouping of systems and subsystems;
- the state of each component can be made evident to the student through the use of color, icons, or other mechanisms;
- changes in each component's inputs and outputs can be made evident to the student;
- the student should have full access to the model through simulated controls, indicators, test points, and replacement operations; and
- provision should be made for the instructional system to configure the simulation in any normal or faulted mode.

Physical Simulation

Maintenance concepts and procedures have imaginal as well as conceptual aspects. The use of qualitative simulation does not, therefore, imply that training should not treat the physical characteristics of the equipment or procedures. Needed, therefore, is a physical simulation that reflects the actual appearance of the equipment. Two considerations drive the design of the physical simulation.

First, physical depictions of the equipment must be tied to corresponding qualitative depictions.

Second, just as the conceptual structure of the equipment is reflected in the qualitative simulation, its physical structure should be represented in its physical simulation. Views of the equipment's main assemblies and subassemblies should be constructed to reflect the access paths to particular components. The means for simulating assembly and disassembly of the equipment should be provided.

Conveying Functionality

Along with structural and physical aspects of maintenance training, we have pointed out the importance of information relating the function of devices. Technicians can rely on functional information -- about the function of the device itself and about the function of its subsystems and components -- in most or all of the reasoning tasks required for maintenance.

Functional knowledge of a device, its subsystems, and its components is used primarily in maintenance to determine whether or not the device, subsystem, or component is functioning properly. This implies that, available within the qualitative simulation, should be a characterization of the function of each element so that students can be asked or informed about the functional status of the element. Such a characterization could describe how the system functions under normal operating conditions, indicate the range of acceptable outputs, and contain contextual information such as the elements with immediate connection to the one in question.

For other reasoning tasks, the design or goal structure of the equipment may be of use. If for example, a subsystem cannot be restored to a fully operational state, the maintainer can use information describing the purpose of the subsystem to decide on the most effective partial repair. Material should be incorporated into the training system that indicates the role of each component and subsystem in the goal structure(s) employing that component or subsystem.

Representing Procedures

Another component in the infrastructure of automated maintenance training is the computational representation of maintenance procedures. From the discussion in Section 1.2.2, we can derive the following elements:

- a description of the procedure's control structure, that is, its steps and choice points,
- how the procedure employs and manipulates the mental and physical models of the device, and

o the functionality or goal structure of the procedure.

It almost goes without saying that the representation should be executable. That is, an interpreter should be constructed that can execute the procedure in conjunction with a particular configuration of the mental and physical model.

Any number of formalisms can be used to satisfy these requirements. Perhaps the leading contenders are ATN grammars, production systems, and and-or graphs. The use of one formalism does not exclude others, and in some cases mixtures, combinations, and redundant representations may be useful.

Troubleshooting Expertise

Effective troubleshooting practice in both automated and traditional environments depends critically on the availability of troubleshooting expertise. For the purposes of automated training, this troubleshooting expert should reflect the nature of human troubleshooting skills. Thus, something on the order of Hunt and Rouse's (1984) fuzzy rule-based model would be appropriate. The essential features of this model are rules that implement both knowledge-based and context-free troubleshooting methods and an explicit representation of the course of the troubleshooting process. Thus, the model could be used to exhibit each step in troubleshooting, its rationale for taking the step, and changes to the problem state after the step has been taken. Hunt and Rouse's model suffers, along with others, from a psychologically unrealistic model of the evaluation of the utility of alternative observations and a simplified mental model of the equipment. It, nonetheless, contains the major features of an instructionally useful model for troubleshooting expertise.

(2) Instructional Methods

With at least a vague conception of the infrastructure for interactive maintenance training, we can be more specific about the instructional paradigms and curricula. A top-level approach to curriculum design might divide the course into three sections, corresponding to the three main instructional objectives described in Sections 1.2 and 1.3: mental models, procedures, and troubleshooting, and also, incidentally, reflecting the structure of more traditional maintenance training.

Some specifics of each major course section are given below. In treating each section, we briefly describe some of the exercises that might be used, and we provide guidelines for curriculum design within each section.

Teaching a Mental Model

1. Physical and Conceptual Structure. Students are shown images of the physical equipment and asked to identify

individual components, their function, and their immediate connections.

2. Causal Reasoning. Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode.

3. Functional Reasoning (a). Students are shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state.

4. Functional Reasoning (b). Students are shown the actual outputs and inputs to an element and asked to determine whether or not the element is faulted.

5. Physical and conceptual appearance. Students are asked to discriminate among component states on the basis of some physical depiction of those states.

The exact sequence of these exercises should be designed to reflect and convey the overall structure of the equipment. In the typical case, where the equipment can be hierarchically decomposed, the exercises can traverse this decomposition in a depth-first fashion so that students learn to reason about a subsystem immediately after learning to reason about each of its components.

These exercises should also be implemented with a view to whole-task training. Many, if not all, of them could be embedded in mini-troubleshooting problems in order to illustrate the application of qualitative reasoning to troubleshooting.

Teaching Procedures

1. Operation. Students are required to perform certain operational functions using both a physical and conceptual simulator. That is, each step in the procedure must be executed within the physical simulator and the conceptual simulator. For complex procedures the goal structure of the procedure should be tracked during procedure execution.

2. Calibration. Students work with a physical simulation of the device to practice required calibration and adjustment tasks. A conceptual simulation of the system being adjusted or calibrated shows relations among the components involved in the process.

3. Testing. Students are required to carry out fixed testing procedures on a physical simulation of the equipment. A conceptual simulation of the components being tested is used

to exhibit or query the student on the states of these components.

4. Access and Disassembly. Students are given the task of gaining access to a particular component. They use a physical simulation of the device to practice the task. A matching conceptual simulation shows which components are accessible at each point in the procedure.

Curriculum design for procedure training is difficult because of subtask relations among procedures often violate the natural coherence relations. The best recommendation to be made in this regard is that subprocedures should be taught before their procedures and that related procedures should be taught in succession. Thus, for example, if a particular test requires disassembly of the equipment, the disassembly procedure should be addressed before the testing procedure. Furthermore, all disassembly procedures should be submitted to a structural analysis that reveals how different branches provide access to different components. The results of this analysis should be reflected in the curriculum so that students are taught how to access (structurally) neighboring components together.

5. Procedure Selection and Use of Job Aids. Students are asked to identify the procedures needed to deal with particular situations and to select any appropriate job aids. Support is provided for this exercise in the form of subgoals and intermediate steps needed to arrive at the proper selection.

Curriculum design for job aids is (or should be) a simpler matter since the curriculum can be designed to reflect the structure of the aids themselves. Even if the aids are not designed systematically, the instructional designer should develop a procedure for selecting the correct aid and design the curriculum to reflect the structure of that procedure.

Considerations pertinent to whole-task training should be given to all (1-5) of the above methods for procedure training. Some of the procedures addressed by this training constitute whole tasks in and of themselves. Others (e.g., disassembly) are enlisted in the service of superordinate tasks. Whole task training can be partially implemented using an apprentice model in which the student observes an automated expert engaged in some difficult task (say troubleshooting) and practices component procedures, testing, for example, as they arise in the course of the task.

6. Redesign and Jury Rigs. Students are provided with conceptual simulations of tasks requiring complete or partial reconstruction of the equipment. For example, students could be required to restore as much functionality as possible with a limited inventory of spare parts or with other constraints on the reconstruction.

Lessons employing exercises of Type 6 should be arranged to traverse the major systems and subsystems in a systematic (depth first) fashion.

Teaching Troubleshooting

1. Troubleshooting. Students are provided with a conceptual simulation containing a single faulted component. At each point in the troubleshooting exercise, students would choose an action and exhibit the consequences of the action. The exercise could take many forms. For example, students might be prompted to select actions diagnostic of a particular fault or sets of faults. Other forms of troubleshooting practice can be found in Brown, Burton, and de Kleer (1982).

2. Reverse Troubleshooting. Students are told that a particular component is faulted. They are required to predict the results of certain observations based on this information. Causal reasoning patterns can be elicited or exhibited during the course of these exercises.

3. Case Studies. Students could be given real case studies of intractable troubleshooting problems. Computer support could be provided for collaborative problem solving and for peer and expert critiques of proposed solutions.

A typical troubleshooting curriculum might have the following lessons.

1. A set of reverse troubleshooting and troubleshooting problems that cover the major topological patterns found in the device. Each pattern would be addressed first by reverse troubleshooting exercises and then by troubleshooting exercises.

2. A set of reverse troubleshooting and troubleshooting problems that cover the equipment's mission-critical faults and their nearest neighbor. Students would first reverse troubleshoot each major fault and its neighbor and then troubleshoot the pair.

3. A repetition of Lesson 1 without reverse troubleshooting.

4. A repetition of Lesson 2 without reverse troubleshooting.

5. A mixture of Lessons 3 and 4.

Reverse troubleshooting in this curriculum plays the role of a cognitive support which is gradually faded from the curriculum. Other cognitive supports (e.g. external hypothesis lists) should also be withdrawn in the last lesson.

(3) Instructional Support

In implementing computer-based training it is important not to lose sight of certain critical functions normally provided by instructors in traditional classroom settings. Of particular concern are functions that establish overall goals, motivation, and coherence to the effort. Instruction of this type is related to what Gagné and Merrill (1990) call the enterprise addressed by the instruction. Of equal importance are the tutoring and coaching functions whereby instruction is adapted to the moment-to-moment needs of individual students.

Enterprise-Oriented Instruction

Traditional classroom methods may play a role in establishing a student's sense of the maintenance enterprise, but in interactive environments, enterprise-oriented instruction should perhaps be viewed more in terms of arts and entertainment (taken seriously) than traditional instructional methods. Thus, mechanisms such as video clips, computer games, and special effects may be appropriate vehicles for conveying the overall significance of maintenance skills.

Tutoring in Interactive Environments

Some tutoring capabilities have been offered in maintenance training (e.g., Brown, Burton, and de Kleer, 1982; Towne and Munro, 1988; Towne et al., 1990), and some general techniques have been developed that might have a place in maintenance training. Some approaches to computer-based tutoring and coaching (e.g., Anderson, Boyle, and Reiser, 1985) attempt to derive a student's particular understanding (in an information-processing sense) of an exercise (including the nature of impasses and misconceptions) and to direct advice to that particular understanding. Others (e.g., Burton and Brown, 1982) make a more global evaluation and direct advice to the student whenever specific deviations from optimal behavior are observed. Both types are appropriate to the type of maintenance training suggested here. Since the former requires considerable investigation of intermediate states of learning, the latter are more easily implemented.

5.2.3.2 Automating the Instructional Design Process

What follows is a treatment of each of the major components of the training system described above with a view to determining which aspects of its development are amenable to automation.

(1) Mental and Physical Simulations

Since the instruction described is based on mental and physical representations of the equipment to be maintained, Halff would first ask what sorts of computational machinery is available for these representations.

Representation of qualitative reasoning structures and practices has received considerable attention in the literature and a number of computational approaches are available for representing mental models. In addition, as was mentioned above, it may be possible to use a quantitative model of the equipment and provide that mathematical model with the conceptual interface of a mental model.

However, a signal shortcoming of available mechanisms for representing mental models is the lack of an approach to the problem of chunking. All current approaches either model the equipment as a flat network of components or rely on the model's designer to provide a hierarchical decomposition. Lacking any progress in automatic chunking of mental models, the division of a device into meaningful systems and subsystems will remain the responsibility of the instructional developer.

Computer representation of the physical aspects of a device appear to be less of a problem than representation of its conceptual structure. The maintenance training community has considerable experience with what is known as "2-D simulation," and a number of systems appear to offer adequate power, typically through the use of videodisc and computer graphics. It should be noted that little in the way of knowledge representation is provided with these systems, thus precluding any reasoning about the physical structure of the device.

Perhaps the most complete and interesting effort on modeling of equipment for maintenance training is the IMTS (Towne & Munro, 1988; Towne et al., 1990). Developers using the IMTS can create a qualitative simulation of a device along the lines suggested earlier. In addition, the IMTS provides an interface to an older type of training simulator, the General Maintenance Training Simulator (GMTS) (Towne, 1986), that provides a physical representation of the device under maintenance. The combination of these two devices offers all the representational power that is needed for a wide range of devices.

(2) Representation of Procedures

Like devices, the representation of procedures has attracted considerable attention in the cognitive science community. As was mentioned above, a number of devices are available for representing procedures. Unfortunately, no one has found it profitable to provide a system devoted to the representation of maintenance procedures. The hardest part of developing such a system, however, would probably be design of the knowledge structures that it employs and manipulates. These knowledge structures are nothing more than the mental and physical models which put the prospect of a viable system for representing maintenance procedures well within reach.

(3) Building an Automated Troubleshooting Expert

The automated troubleshooting expert must possess both context-independent and context-specific skills. The context-independent skills, by definition, will be common to all maintenance courses and, therefore, need not concern the developer of any particular course. Knowledge-based, context-specific strategies vary from equipment to equipment and do, therefore, concern instructional developers. The best approach to deriving knowledge-based troubleshooting skills is a machine learning mechanism that could derive them automatically in simulated troubleshooting exercises. Since the creation of such a mechanism would be a major research project in its own right, for practical purposes knowledge-based methods would have to be formulated by subject matter experts for each equipment.

(4) Exercise and Curriculum Development

Computers show considerable promise as devices for generating curricula of exercises and examples. The instructional approach described above relies heavily on such curricula, so that it behooves us to ask how computers might assist in their generation. Needed are:

- a template for the curriculum itself or part of the curriculum to be generated by the computer,
- frames for representing exercises in such a way that they can be fit to the template, and
- a search mechanism for filling the template with particular exercises.

In the absence of precise specifications for any of the lessons suggested above, curriculum templates would employ a special-purpose programming language. The frames for representing individual exercises might contain slots for content, procedure, prerequisites, subtasks, and cognitive support. The search mechanism is essentially an implementation problem and will not be considered here.

(5) Instructional Support

Two kinds of instructional support for the training activities were suggested earlier: enterprise-oriented instruction that provides coherence, motivation, and a sense of the overall significance of the maintenance task; and tutorial functions that provide advice appropriate to a student's moment-to-moment situation during the course of an exercise.

Halff does not consider the automated generation of enterprise-oriented instruction to be feasible. Thus, any system that serves as a vehicle for maintenance training should provide for the inclusion of human-generated materials of the sort needed to

maintain the cognitive integrity of the enterprise under instruction.

By contrast, it is entirely feasible to provide for some forms of automated tutoring with no extra effort on the part of instructional developers. A case in point is the IMTS, which evaluates troubleshooting action against an optimal troubleshooting model so that it can suggest more fruitful approaches at appropriate times. The optimal model used by IMTS suffers from the fact that it is more of a competence model than a performance model, and no attempt is made within IMTS to derive an information-processing account of student actions. However, Halff sees no intrinsic difficulties in improving the tutor along these or any of a number of other lines. Some of these improvements, such as the use of empirical bug catalogs in diagnosing student problems, might entail considerable extra effort on the developer's part. Others, however, such as a more psychologically valid method for choosing troubleshooting actions, would require very little extra effort on the developer's part.

5.3 Teaching Troubleshooting Procedures (TTP)

In this section Halff extends the ideas presented in Section 5.2 in two respects.

First, in this section Halff focuses on procedure learning and, in particular, on learning a class of standard troubleshooting procedures based on fault trees. This class of methods reduces troubleshooting tasks from the problem-solving activity described in Section 5.2 to one of implementing a general procedure to deal with particular malfunctions. The data needed to implement the procedure for a particular malfunction can be found in the troubleshooting sections of the equipment's technical documentation.

Second, it is more specific to the training methods and requirements for one particular troubleshooting procedure, namely, that of repairing an Air Force T-38A aircraft that fails to start on the ground. This procedure is described in U. S. Air Force (1989).

In this approach to instructional design, Halff first describes the content to be taught, organizing the discussion around the no-start troubleshooting procedure. He then derives a set of instructional objectives. Finally, relying on recommendations in Halff (1990), he draws out major implications for instructional design and practice.

5.3.1 Instructional Content: The T-38A Maintenance Scenario

5.3.1.1 The T-38A Starting System

Since the malfunction used as an example here is that of a T-38A that fails to start, understanding the troubleshooting task obviously requires some understanding of the starting system of the T-38A.

The T-38A is a two-engine jet aircraft. Each engine must be started independently, and the right engine is started first. In this discussion of the no-start troubleshooting procedure and related issues, Halff assumes that the procedure addresses the right engine.

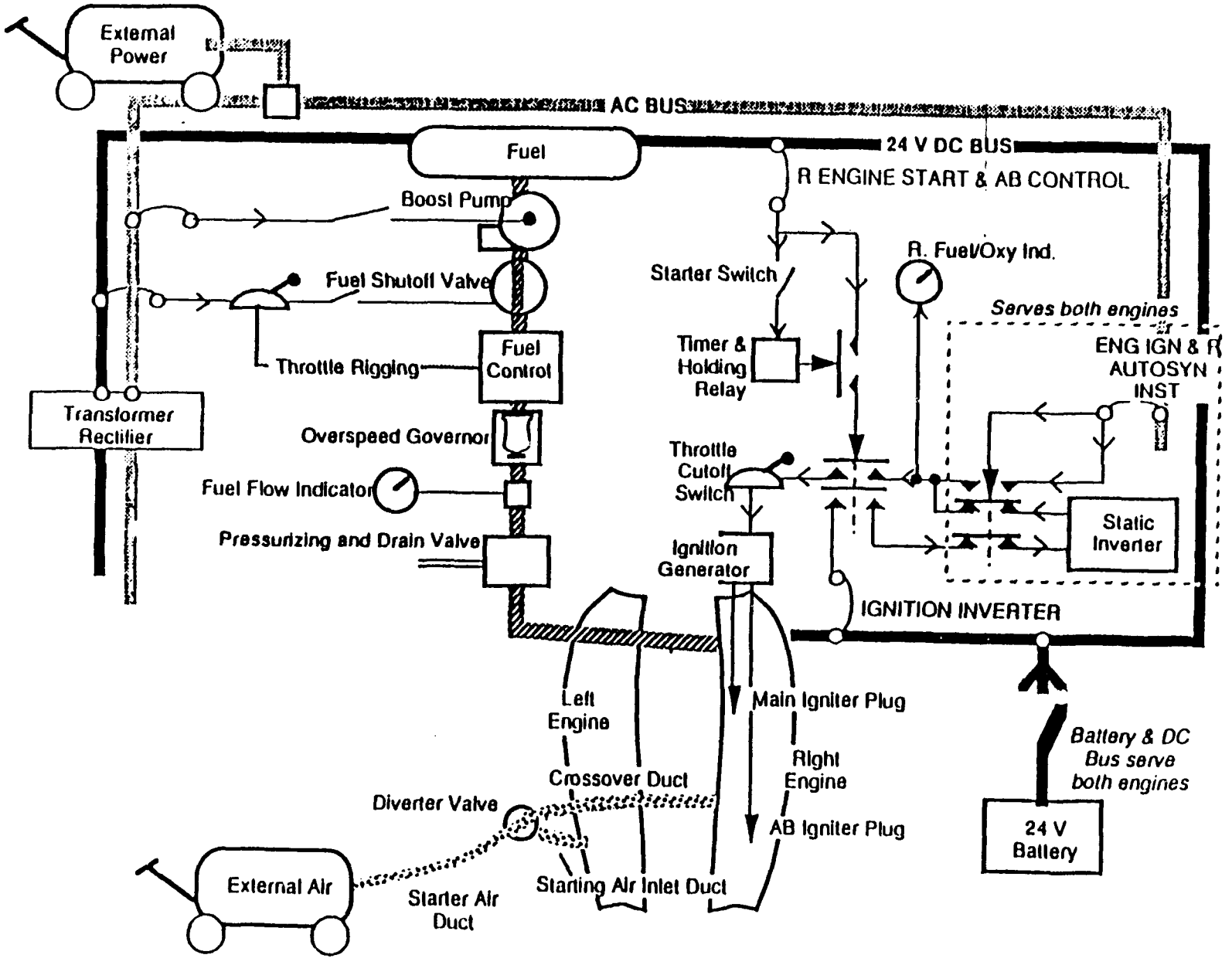


Figure 5.1 T-38 Supersonic Trainer Engine Start System.

Figure 5.1 is a grossly oversimplified schematic view of the starting system of the T-38A. It contains only those components that figure significantly in the no-fault troubleshooting procedure.

Needed to start an engine are compressed **air** (used to air-motor or rotate the engine), **fuel**, and **ignition**. On the ground, air is supplied by an external compressor attached through an air hose to a diverter valve on the aircraft. The diverter valve is positioned by the ground crew to direct air to the engine (left or right) being started.

Fuel for each engine is supplied through fuel lines controlled by the throttle. An additional control over fuel flow is an overspeed governor that is subject to leaks which can impede the fuel needed for a normal start. A boost pump, designed to supplement gravity feed at high altitudes, can be used to purge air from the fuel line.

Ignition for each engine is supplied through two igniters, a main engine igniter and an afterburner (AB) igniter. Each of these igniters provides a train of sparks at a steady rate (3 sparks every 2 seconds).

The delivery of air, fuel, and ignition to the engine takes place under partial control of an electrical starting system, the main elements of which are shown in Figure 5.1. The pilot starts the engine by depressing a starter switch and advancing the throttle to its IDLE position. Depressing the starter switch engages a timer and holding relay that, in turn, arms an ignition circuit for about 30 seconds. Once the circuit is armed, moving the throttle to IDLE closes a throttle cutoff switch and thereby causes the igniters to fire. Advancing the throttle also delivers fuel to the engine.

A final important element in the starting system is the delivery of electrical power to the igniters and control circuits. DC power is needed for the ignition control circuit (i.e., the timer and holding relay); AC power is needed to fire the ignition. On the ground, power can be supplied by two sources: a (DC) battery in the aircraft and an external AC power source. The battery is controlled by a switch which must be turned off if external power is used. If the aircraft is started under battery power, a static inverter is used to convert this power to AC for the igniters. If the battery is not used to supply DC power, a transformer-rectifier on the aircraft converts AC power to DC. Also of importance to troubleshooting is the fact that the static-inverter circuit powers a subset of the right-engine cockpit instruments and, in particular, the right-engine fuel/oxygen indicator. After starting, electricity (AC) is supplied by generators (not shown in Figure 5.1) powered by each engine. These generators begin to operate when the engines reach a set cut-in speed.

5.3.1.2 Starting the T-38A

With this background, the starting procedure for the T-38A should be clear. To paraphrase the Flight Manual,

1. Make sure nothing is in the way of the aircraft.

2. Set the diverter to the right engine.

3. Apply compressed air to rotate the engine.

4. When the engine speed reaches 14% RPM (or at least 12% RPM), push the right starter switch.

5. Advance the throttle to idle.

6. Wait for the engine to start.

Ignition should occur before fuel flow reaches 360 lb/hr. If not, turn the throttle off, maintain air flow for 2 minutes to evacuate fuel from the engine, and restart.

The EGT (engine thrust) should begin to rise within 12 seconds of start of fuel flow. If it does not, abort the start.

The generator should cut in before the 30 sec. ignition circuit times out. If this does not occur, check to make sure that the engine light is Normal. If it is, push the start button again to provide electrical power for the start.

7. Check the engine instruments, the hydraulic power, and the caution light panel.

8. Repeat steps 2-7 for the left engine.

9. Disconnect the air supply and, if connected, the external power.

10. Make sure the battery is switched on.

5.3.1.3 Possible Faults

Any of a vast number of faults could cause the T-38A engine not to start. Halff's analysis, however, is limited to the small number of faults actually listed in Table 5.1. The reader will note that, in many cases, identification of a fault is not that specific; a fault is any determination that terminates the no-start troubleshooting procedure. By the same token, what appears in the Repair column of Table 5.1 is, in some instances, a repair, and, in others, directions for further troubleshooting.

5.3.1.4 The Troubleshooting Procedure

Figure 5.2 shows the fault tree for the no-start troubleshooting procedure. Notice that the terminals of the tree correspond to the faults identified in Table 5.1. Also notice that some of the terminals are marked as "Last Resort." In the troubleshooting procedure, each such fault is assumed as a last resort when all other faults on its branch have been eliminated.

The fault tree is nothing more than a description of the troubleshooting procedure's structure. Needed to convert that description into an actual procedure is a fault-tree interpretation procedure or schema. Table 5.2 contains a description of this schema in pseudocode. A more colloquial description is as follows. To troubleshoot any component, first check its overall functionality. If the component is nonfunctional, then either repair it, if it is a terminal in the fault tree, or troubleshoot its subcomponents. However, any last-resort fault (that is, one isolated solely by eliminating other possibilities) should be repaired without further ado.

A critical aspect of this fault-tree approach is the structure of the tree itself. In particular, the structure of the tree (a) reflects the structure of the equipment itself as a hierarchy of subsystems and (b) is predicated, by virtue of Line 2.1, on the availability of tests of the functions of all components except last resorts.

Needed to instantiate the schema for the no-start troubleshooting procedure are Table 5.1, Figure 5.3, and the observations needed to perform the tests called for in Line 2.1. These observation procedures, are listed in Table 5.3.

A fault tree is a hierarchical structure typical of those found in most highly developed cognitive skills to simplify overly complex procedures and thus render them amenable to skilled execution (Chase and Ericsson, 1982). A fault-tree representation of a troubleshooting procedure offers cognitive benefits that outweigh any inefficiency that it may introduce into the troubleshooting process. Using more efficient but less structured approaches would inevitably result in slower learning and in lower terminal speed and accuracy of performance. The instructional design proposed below is predicated on the assumption that a fault-tree representation of the no-fault troubleshooting procedure (and other procedures of the same sort) is the most appropriate representation for instructional purposes.

Table 5.1 FAULTS CAUSING NO-START CONDITION.

Fault	Description	Repair
Ignition Faults		
Ignition Circuit Breakers Not Engaged.	Three circuit breakers are present in the ignition control circuitry: R ENGINE START & AB CONTROL, ENGINE IGNITION & R AUTOSYN INST. and IGNITION INVERTER. Any or all of these could be improperly engaged.	
Defective Static Inverter.	The static inverter can fail to supply AC power to the igniters.	Replace the static inverter.
Defective Electrical System.	The engine's electrical system can be faulted in such a way that even external AC power is not being delivered to the igniters.	Troubleshoot the electrical system using procedures defined in the Electrical Systems Manual.
Defective Igniters.	The igniters themselves may not fire, even when properly powered.	Remove the engine.
Fuel System Faults		
Defective Aircraft Throttle Rigging	The throttle rigging connects the throttle to the fuel control system. If defective, fuel flow may not be adequate for starting.	
Fuel System Circuit Breakers Not Engaged.	The fuel control system will not operate if its circuit breakers are not properly engaged.	
Air in System.	Air in the fuel system may retard fuel flow.	Apply external power and start engine with boost pumps on.
Altitude Problem.	Fuel feed may not achieve required pressure at high altitudes.	
Fuel Shutoff Valve Closed	A fuel shutoff valve must be open to permit the flow of fuel.	
Excessive Drain from Engine Components	Fuel may be draining off through engine drain lines.	
Internal Leakage from Engine Drain Indicator.	Excessive amounts of fuel are drained off through the bypass hose of the overspeed governor.	Replace the overspeed governor.
Unknown Fuel System Fault.	Fuel may not flow for reasons other than those listed above.	Remove engine.
Starting System Faults		
Defective Ignition Time-Delay Relay	Relay is not engaged for 30 sec upon depressing starter, thus preventing electrical power from reaching the igniters.	
Defective Diverter Valve	The diverter valve is not positioned properly.	
Blocked Air Duct	The hose from the compressor to the aircraft may be blocked or kinked, thus preventing compressed air from reaching the engine.	
Defective Engine Starting Air Inlet Duct and/or Crossover Duct	Ducts from the diverter to the engines may not allow passage of air.	
Engine Seized or Binding	The engine may not be rotating freely.	Remove engine and determine cause of problem.
Operational Problems		
Altitude Problem	Engine may not start at high altitudes.	

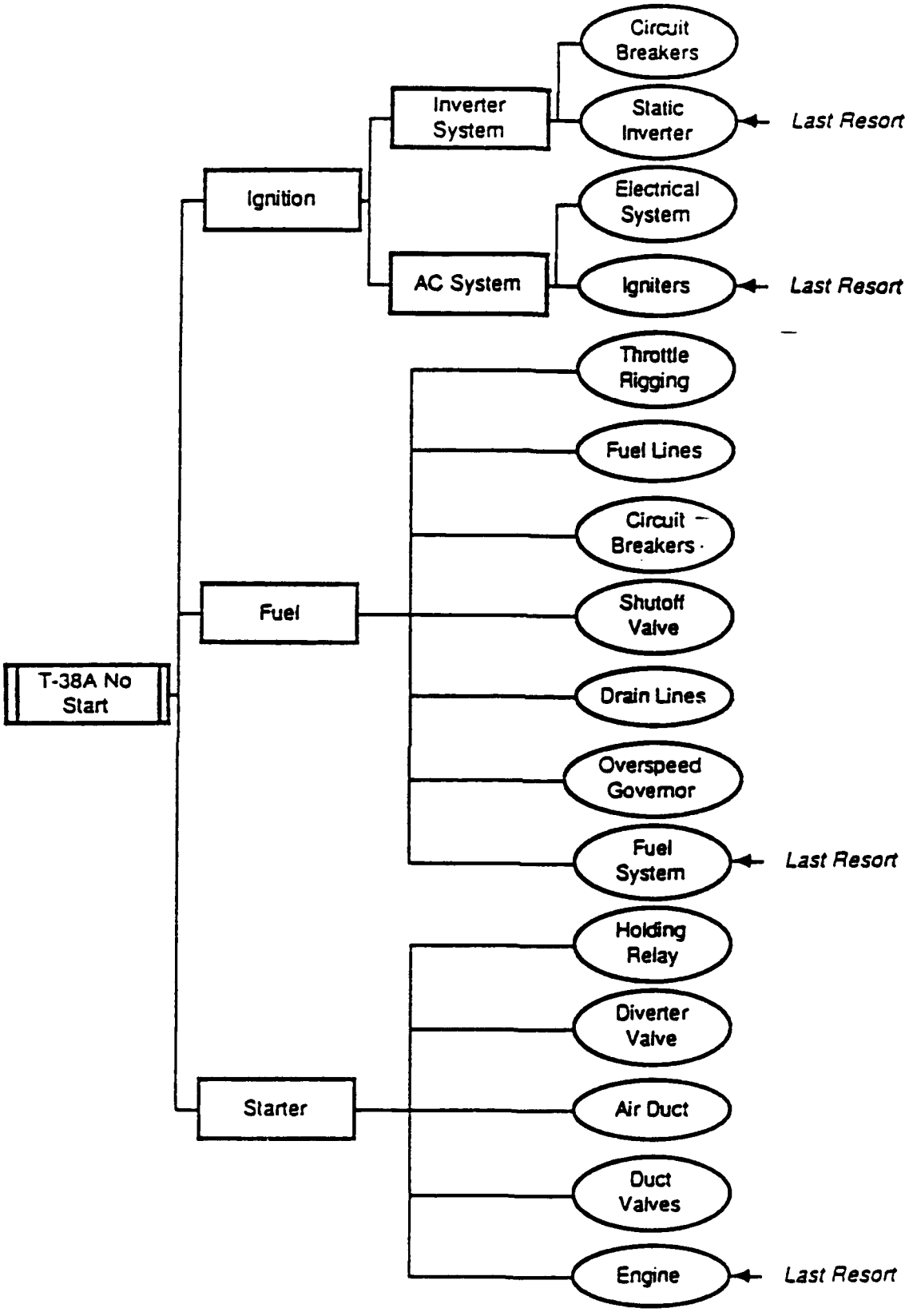


Figure 5.2 T-38A No-Start Fault Tree.

Table 5.2 FAULT-TREE INTERPRETATION SCHEMA.

```
To troubleshoot a component
    1.0 If the component is a Last Resort then
        1.1 Repair the component
        1.2 Conclude[a] that the component was faulted
    2.0 Else (component is not a last resort)
        2.1 Check the overall functionality of the component
        2.2 If the component is OK then
            2.2.1 Conclude that the component is not faulted
        2.3 Else (component is not OK)
            2.3.1 If the component has no subcomponents
                2.3.1.1 Repair the component
                2.3.1.2 Conclude that the component was faulted
            2.3.2 Else (component has subcomponents)
                2.3.2.1 For each subcomponent of the component
                    2.3.2.1.1 Troubleshoot the subcomponent
                    2.3.2.1.2 If the subcomponent was faulted then
                        2.3.2.1.2.1 Conclude the component was faulted
                    2.3.2.1.3 End If
                2.3.2.2 End For
                2.3.2:3 Conclude fault is unknown
            2.3.3 End if
        2.4 End if
    3.0 End If
```

Note: [a]"Conclude" statements are equivalent to function returns in that they terminate the troubleshooting procedure and return a value describing the fault thus isolated.

5.3.2 **Instructional Objectives**

Since much of this effort relies on the general scheme described in Section 5.2, we need to frame the particular content described above in terms of the general objectives described there. Recall that these instructional objectives consist of (1) a mental model of the equipment, (2) the procedures used to maintain the equipment, and (3) the special skills needed for troubleshooting as problem solving.

5.3.2.1 Mental Models

Mental models of equipment, as described in Section 5.2, have three aspects structure, function, and imagery.

Table 5.3 FUNCTIONALITY TESTS FOR FAULT TREE COMPONENTS.

Component	Test
T-38A No Start	Standard starting procedure
Ignition	Observe AB and main plug firing during ignition without external power
Inverter System	Check Fuel/Oxy Indicator without external power
Circuit Breakers	Check engagement
Static Inverter	Last Resort[a]
AC System	Observe AB and main plug firing with external power
Electrical System	Check voltage at Pin N in engine ignition and accessories disconnect plug[b]
Igniters	Last Resort
Fuel	Check for fuel mist in exhaust
Throttle rigging	Advance throttle past idle and check for fuel flow or ignition
Fuel lines	Attempt repair by running boost pumps to clear air
Circuit breakers	Check for engagement
Shutoff Valve	Observes status
Engine Drain Lines	Check for excessive fuel drainage
Overspeed governor	Check flow rate through bypass hose
Fuel System	Last Resort
Starter	Engine achieves 14% RPM with proper air flow
Holding Relay	Check engagement for 30 sec.
Diverter valve	Inspect position
Air Duct	Check for kinks, obstruction
Duct valves	Check position
Engine	Last resort

Note: [a]Last-Resort components are always deemed to be faulted by a process of elimination.
[b]Not shown in Figure 1.

(1) Device Structure

The structure of a device is a formal qualitative description of that device, often called a qualitative model. Such a model describes the behavior of individual components, that is, how they change state in response to changes in input and how their outputs vary according to state. It also describes the behavior of the device as a whole, that is, how the outputs of one device are connected to the inputs of another.

Fault trees of the type described above are closely related to qualitative models of equipment in that the qualitative model, therefore, provides cognitive support for the troubleshooting procedure. For example, inspection of the right engine fuel/oxygen indicator for evidence that the static inverter system is functioning correctly only makes sense within the context framed by a mental model of the ignition system as shown in Figure 5.2.

A qualitative model of the entire starting system could be (indeed has been) created from a conception like that shown in Figure 5.2 and this figure would make a reasonable basis for a qualitative model if our only concern was the no-start troubleshooting procedure. However, any realistic maintenance course must also address many other malfunctions and troubleshooting procedures. The model shown in Figure 5.2 is not really a system, but rather the components of several systems that, taken together, are responsible for starting the aircraft or failing to start it. A different function (e.g., establishing radio communications) or even a different malfunction of the same system (e.g., failure to start in flight) would involve different components of the same or different systems. If troubleshooting training were to be based on models like that of Figure 5.2, a different model would be needed for each malfunction in the curriculum. The proliferation of such models would easily defeat any advantage of model-based training.

Needed, therefore, is one or a small number of models that can support troubleshooting training for all malfunctions of interest. Function or malfunction-specific models like that of Figure 5.2 may still have a place in training and may even take a measure of cognitive reality. However, a more general scheme is needed to define the mental models that constitute instructional objectives in this context.

The design of such a scheme for the T-38A is beyond the scope of this paper. Nonetheless, we can describe a strategy for constructing a mental model of the aircraft. This strategy has a bottom-up and a top-down component. The bottom-up component calls for a fault-tree analysis of each malfunction covered in the Engine Conditioning Manual, the Electrical Systems Manual, and other such documents. One result of this analysis will be a collection of tables like Table 5.1 and Table 5.3, specifying components and elementary procedures for repair and observation.

This information can be used (along with other technical documentation) to construct component models that reflect normal and faulted behavior and that can be appropriately manipulated during troubleshooting.

The top-down component of the strategy calls for analysis of the aircraft's systems using the scheme laid out in the Flight Manual. As illustrated in Figure 5.4, this scheme describes the major systems of the aircraft: fuel, fuel control, electrical, etc., together with their subsystems.

Combining the bottom-up and top-down analyses is a matter of merging the primarily component-wise information in the former with the topological and structural information in the latter.

(2) Device Function

For purposes of troubleshooting, device function is best thought of in terms of operating functions and malfunctions. Section VI of the T-38A Engine Conditioning Manual is an excellent starting point for such an analysis. Identified in that section (entitled "Troubleshooting Engine and Related Problems") are a number of operating modes of the aircraft. For each operating mode is a list of malfunctions. For each malfunction, a troubleshooting procedure (theoretically based on a fault tree) is provided which identifies particular components and subcomponents. Thus, it is possible to construct a functional hierarchy based on operations, potential malfunctions, and the normal or faulted operation of individual components. This hierarchy for T-38A engine functions is shown in Figure 5.5.

This functional hierarchy is not the same as the structural breakdown by system shown in Figure 5.4, particularly since several systems are usually involved in each operation. Starting Operation, for example, involves the starter system, the fuel system, the fuel control system, the ignition system, the electrical power system, and the engine. The functional hierarchy of Figure 5.5 is organized around the T-38's functions or missions, and it comprises a distinct, important aspect of the instructional objectives. In Section 5.3.2.2, we will discuss the part that it plays in curriculum design.

(3) Imagery

Imagery, as described in Section 5.2, is the perceptual face of the technician's conception of the equipment. As such, it covers the appearance (in sight, sound, and other senses) of the equipment, the physical actions that implement observations and repairs, and knowledge of the physical location of components and subsystems.

Direct formal representation of the imaginal aspects of a mental model are well beyond current methods of knowledge representation. What can be constructed, however, is a system of

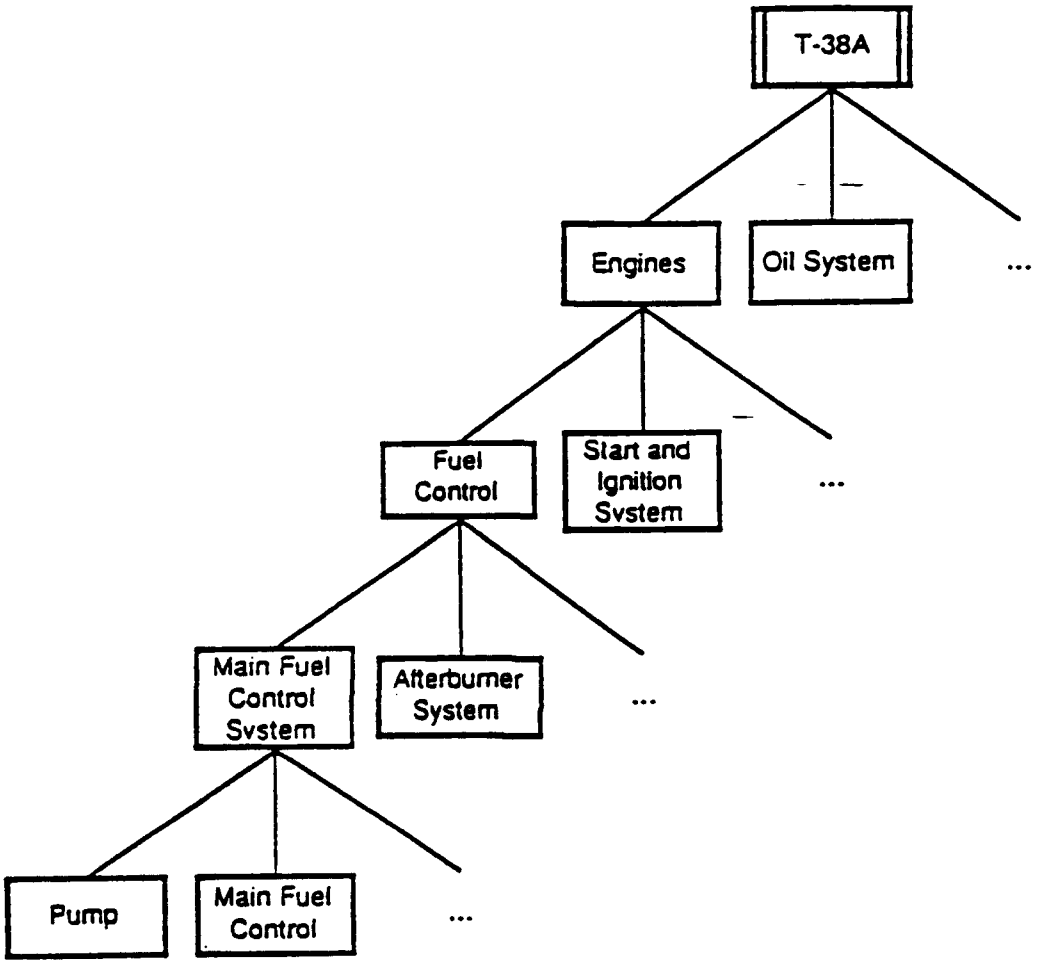


Figure 5.3 Structural breakdown of the T-38A.

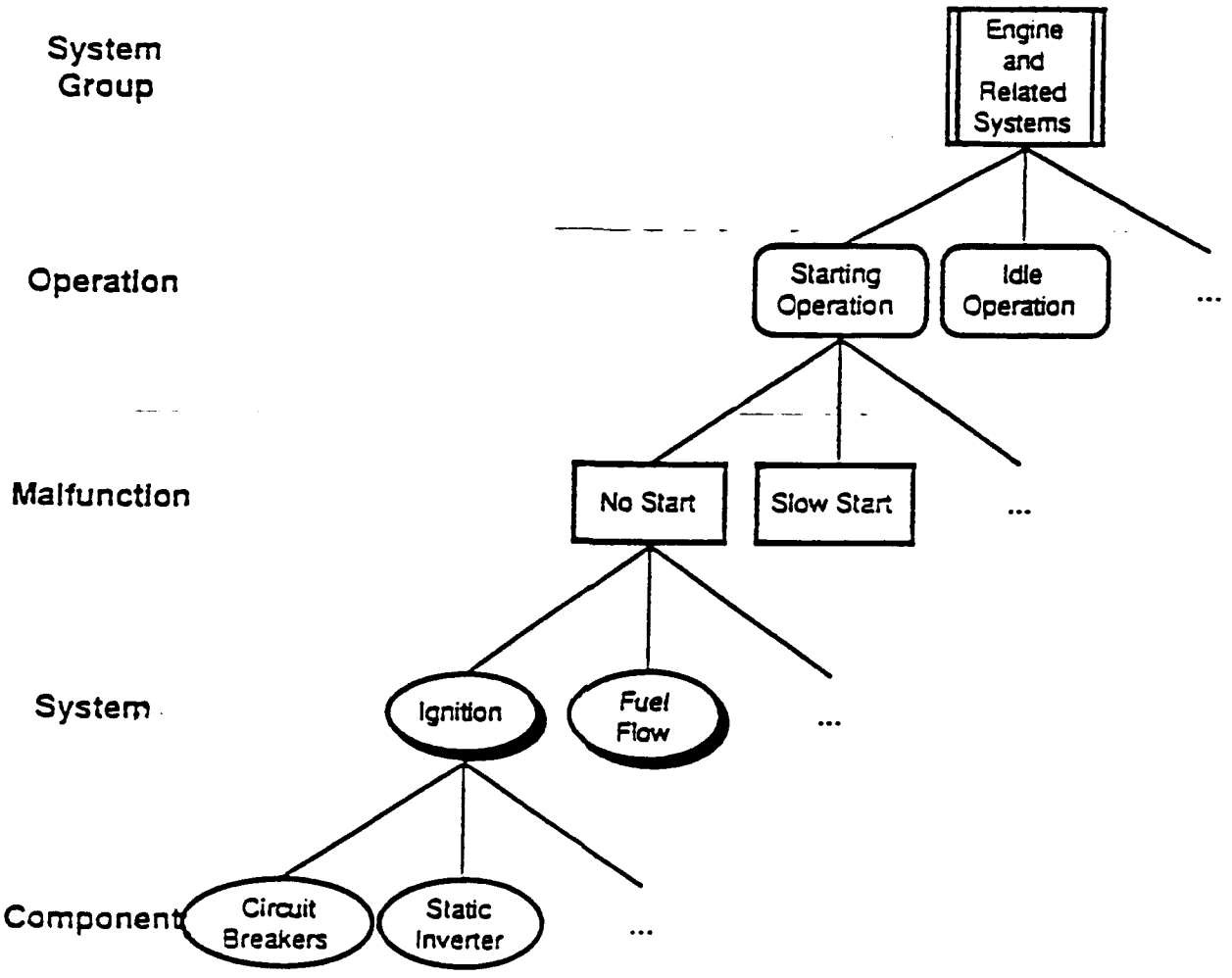


Figure 5.4 Functional breakdown of the T-38A.

tokens that stand for different perceptual chunks. These tokens would include, in the case at hand,

- the perceptual aspects of all observations, such as the sound of the main igniter sparking and the appearance of the fuel-flow indicator;
- the actions needed to effect operations and repairs, such as shorting out the afterburner igniter and engaging a circuit breaker; and
- transitional items needed to refocus attention from one component or operation to another; for example, locating the ignition-system circuit breakers or moving from the cockpit to the rear of the aircraft.

5.3.2.2 The Troubleshooting Procedure

The second type of instructional objective proposed in Section 5.2 is that of procedural knowledge. Some aspects of procedural knowledge representation are discussed there, but a more informed description can be found in Polson and Polson (1990). To a large extent, both of these general discussions are superseded by the specific description of troubleshooting in Section 1.4.

Section 5.3.1.4 suggests that three distinct items of procedural knowledge are involved in mastering the no-start troubleshooting procedure:

- the general troubleshooting schema of Table 5.2,
- the testing and repair procedures listed in Table 5.1 and Table 5.3, and
- the structural information in the fault tree itself (Figure 5.2).

These items constitute three, almost independent, procedural objectives for instruction in the no-fault troubleshooting procedure. By "almost independent" Halff means first that the fault-tree interpretation schema can be learned in such a way that it will apply to any fault tree, not just the one presented in Figure 5.2. Also, the testing and repair procedures can be mastered outside of no-fault troubleshooting procedure and most will probably be useful in other maintenance tasks. Mastery of the fault tree (Figure 5.2) does, however, depend on mastery of the fault-tree interpretation schema.

Representation mechanisms for the types of procedural knowledge of concern here are well discussed in Polson and Polson (1990). Recommended there is the use of Card, Moran, and Newell's (1983) GOMS scheme for representing procedural knowledge. Representational schemes such as GOMS constitute important, but

not complete tools for expressing fault-tree based trouble-shooting procedures.

(1) Representing the Fault-Tree Interpretation Schema

The fault-tree interpretation schema of Table 5.2 is basically a control structure wrapped around retrieval operations that fetch the appropriate elementary repair and operation procedures. The control structure in this schema can be translated in a straight-forward manner directly into GOMS. Not specified in Table 5.2, however, are the operations needed to retrieve information that instantiates the schema. For example, Line 2.1 requires retrieval of a procedure of the sort listed in Table 5.1. Line 2.3.1 requires the retrieval of structural information of the type specified in Figure 5.2. These retrieval operations will vary according to situation and skill level of the technician. Novices will use technical documentation, instructors and other external sources. Intermediate technicians will retrieve this information from long-term memory. In highly-skilled technicians, the schema and its instantiating data will be compiled into a single procedure. A GOMS representation of the schema in Table 5.2 must, therefore, provide different strategies for retrieval operations to reflect the circumstances and skill levels of the technician.

(2) Representing Elementary Procedures

The elementary procedures listed in Table 5.1 and Table 5.3 are eminently suited for representation as GOMS structures. Indeed, a preliminary GOMS analysis of the entire no-start troubleshooting procedure is available as an analysis of the operations in Table 5.1 and Table 5.3.

In a representation of the procedure used by a highly skilled technician, control knowledge and operational knowledge are compiled into one procedure and the fault tree is given no explicit representation at all. Most of the literature on skill development indicates that this composition or compilation is a characteristic of highly automated skills.

(3) Representing the Fault Tree

Fault trees such as that shown in Figure 5.2 are, on the face of it, declarative knowledge, and, in students, they may begin life in declarative form. To be of use, however, their fundamental representation must be procedural. The procedural requirements of the interpretation schema (Table 5.2) are the following functions.

- Retrieve the procedure for checking the functionality of a component (Table 5.2, Line 2.1).
- Retrieve the procedure for repairing a terminal component (Table 5.2, Lines 1.1 and 2.3.1.1).

- Decide whether or not a component is terminal (Table 5.2, Line 2.3.1).

- Retrieve successive subcomponents of a non-terminal component. (Table 5.2, Line 2.3.2.1).

A minimal representation of a fault tree must therefore implement these four functions. For instructional purposes a declarative representation will also be needed together with procedures (perhaps GOMS procedures) for implementing these functions by interpreting the declarative representation.

5.3.2.3 Troubleshooting as Problem Solving

The last category of instructional objective mentioned in Halff (1990) is that of troubleshooting. Halff's concern in including that category was for troubleshooting as a **problem-solving** activity in which the technicians need to discover the appropriate fault isolation strategy. In particular, my reading of the literature in this area suggested a problem-solving procedure with three main components:

- context free rules for isolating faults based on topological patterns,

- device-specific rules for troubleshooting based on known symptom-fault associations, and

- procedures for choosing the most information-laden actions at particular choice points.

This problem-solving approach to troubleshooting relies only on connectivity information in the mental model and a set of device-specific observation-action associations.

By contrast, the focus of this paper is on troubleshooting **procedures**, in which the isolation strategy is provided through documentation and instruction and need only be implemented by the technician.

Troubleshooting as problem solving is an important aspect of skilled troubleshooting and, therefore, of maintenance training, even when a large body of troubleshooting procedures is available. In a system of any complexity, no troubleshooting guide offers complete coverage of the possible faults, and seldom will a guide offer procedures for dealing with complex situations such as test-equipment unreliability, multiple interdependent faults, and intermittent faults.

Schemes are available for representing troubleshooting as problem solving (Hunt and Rouse, 1984; Towne, Johnson, and Corwin, 1983), and instructional systems are available for implementing these schemes.

Of central concern, however, is the relationship between the procedural approach to troubleshooting presented in Section 5.3.1.4 and problem-solving approaches such as those of Hunt and Rouse (1984) and Towne, Johnson, and Corwin (1983). The fault-tree approach, by sacrificing generality, provides the structure needed to render the troubleshooting process manageable with limited time and cognitive resources. Methods for troubleshooting as problem solving have greater generality, but are expensive in terms of both time and resources. A combination could use the fault-tree procedure in the initial stages and invoke a problem-solving process to deal with cases not covered in the fault tree.

Earlier Halff mentioned that the last-resort faults in Figure 5.2 seemed to be guards against the impasse that might occur if a component malfunctions, but each of the tested subcomponents functions properly (see Line 2.3.2.3 of Table 5.2). In addition to guarding the procedure against impasses, these last-resort faults may also play the role of place holders that mark occasions for troubleshooting as problem solving. If so, then the fault tree could be redesigned to eliminate the last-resort faults, and an impasse itself (i.e., Line 2.3.2.3) could signal the occasion for invoking a problem-solving process.

The implications of this suggestion for curriculum design are clear. That is, troubleshooting as problem solving should be taught after, and in conjunction with procedures based on fault trees, and students should be explicitly taught when to enter a problem-solving mode of troubleshooting. These points of transition are the Last Resort items in Figure 5.2 or, alternatively the impasses marked by Line 2.3.2.3 of Table 5.2.

Halff also made brief mention of problem-solving approaches to troubleshooting that select local troubleshooting moves or actions based on device topology, known observation-action associations, and the information value of potential actions. We suggested that students need to be taught these problem-solving techniques and should be trained to invoke them when the fault-tree approach arrives at an impasse.

5.3.3 **Instructional Material and Methods**

In Section 5.2 Halff proposes certain methods for using computers in maintenance training. The discussion is organized around three aspects of a computer-based instructional system.

5.3.3.1 Infrastructure

An instructional infrastructure for computer-based maintenance training consists of computational representations of the instructional objectives. In particular, it contains mechanisms that can simulate and describe the equipment under maintenance in both qualitative and physical terms. It also contains mechanisms that can interpret and describe procedures to be learned. In

addition, an expert troubleshooting system should be available for teaching troubleshooting as problem solving.

(1) Qualitative and Physical Simulation

Recall that the mental model consists of a qualitative model of device behavior, a description of device function, and a collection of tokens that stand for imaginal (perceptual) aspects of the device. The qualitative model can be implemented in a computer using available qualitative simulation systems such as that described in Towne, Munro, Pizzini, Surmon, Coller, and Wogulis (1990). These systems have the following basic capabilities.

- They can instantiate any qualitative model of the sort described in Section 5.2.

- They can carry out qualitative reasoning on the model and thereby determine the states and outputs of specified components based on information regarding states and state-changes in the rest of the device.

- They present to the student a graphical depiction of the system being simulated. This depiction does not generally resemble the physical appearance of the system. Rather, it represents components as icons that can be manipulated by the student and whose appearance reflects the states of the corresponding components. A display from such a system might, for example, bear a passing resemblance to Figure 5.1.

- They provide different views of the system. Presented to the student not just as one view of all components, but rather a collection of views, each showing only the components selected for that view. Some of these views might show the behavior of major subsystems, and thus conform to the breakdown illustrated in Figure 5.4. Others might conform to the functional breakdown illustrated in Figure 5.5 in that, like Figure 5.2, they show all of the components involved in a particular function.

Constructing a qualitative model of the T-38A, then is a matter of entering the qualitative model of the aircraft into a qualitative simulation system, designing the graphical representation of the model, and constructing the views needed for instruction. For troubleshooting training, three types of views are needed.

1. Global subsystem views are needed that present each major subsystem as an independent unit.

2. Views like that in Figure 5.1 should be constructed to show all of the components involved in the functions and malfunctions selected for troubleshooting.

3. A third type of view conjoins the first two and thereby depicts only the components involved in particular lower level nodes of the fault tree. Figure 5.5, for example, provides a view of the Fuel System branch of the fault tree in Figure 5.2.

Needed to support the imaginal aspects of the model is a physical (as opposed to qualitative) simulation of the aircraft. This physical simulation presents to the student, through appropriate media, the sights, sounds, and manipulanda associated with each of the tokens in the imaginal model (see Section 5.3.2.1(3)). These requirements include:

- a simulation of all observations that might be made in the course of troubleshooting, including indicators such as the Right Fuel/Oxygen Indicator and other sorts of observations such as the appearance of fuel mist in the exhaust;

- a simulation of all actions that might be undertaken in the course of troubleshooting, including controls such as the throttle and other actions such as that of shorting out the AB ignition plug; and

- a simulation of transitions that refocus attention from one component of the aircraft to another, including views of panels such as the instrument panels and gross changes of view such as descending from the cockpit and moving to the rear of the aircraft.

As a development strategy, the team developing instruction should walk through every branch of every troubleshooting procedure, noting the views and manipulations needed to support the procedure. These notes can then form the basis for production of the physical simulation in appropriate media.

Naturally, the physical and qualitative simulations should be linked in their implementation so that manipulations of the physical simulation are manifest as state changes in the qualitative simulation, and manipulations in the qualitative simulation are manifest in displays of the physical simulation. For example, if the student, in the qualitative model closes the fuel shutoff valve, then a view of the exhaust in the physical simulation (with the throttle on) should reveal no fuel mist. Conversely, if the student, in the physical simulation, applies external power to the aircraft, then the Ignition Power Transfer Relay should be shown to energize in the qualitative simulation.

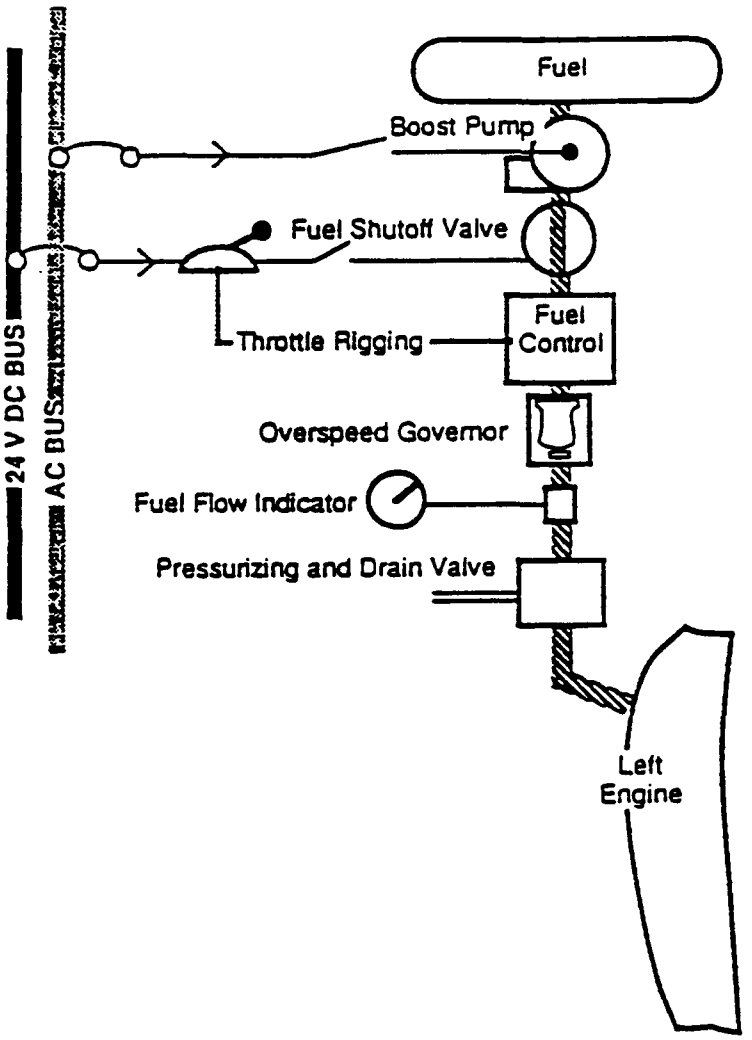


Figure 5.5 Components involved in the Fuel Flow Branch of the No-Start Fault Tree (Figure 5.2).

(2) Procedure Interpretation

A second component of the instructional infrastructure is a computational implementation of the procedures needed for, in our case, troubleshooting. Recall that the formalism used for representing these procedures is the fault tree described in Sections 5.3.1.4 and 5.3.2.2. The components of this formalism are:

- a fault-tree interpretation schema, represented in GOMS or some similar formalism,
- the fault-tree itself, represented in both procedural and declarative forms, and
- the observation and repair procedures attached to nodes in the fault tree, also represented in GOMS or some other appropriate formalism.

Needed to computerize these formal models are an interpreter that can execute the troubleshooting procedure, appropriate links to the simulation models discussed above, and a means of presenting the structure of the procedures themselves to the student. The first two items pose no particular challenges since interpreters are available for this purpose and since the simulations themselves possess the mechanisms needed to link them to a procedure interpreter.

The third item, the means of presenting the procedure itself to the student, is a critical aspect of the instructional design proposed here. Mechanisms are needed for explicit presentation of each of the three parts of the fault-tree approach.

Presenting the Fault-Tree Interpretation Schema

Support for navigating the fault-tree interpretation schema of Table 5.2 should be provided by a display that either indicates the major milestones in the schema or allows the student to indicate these milestones. These milestones include

1. checking the functionality of the component under consideration (Table 5.2, Line 2.1),
2. isolation and repair of a component (Table 5.2, Line 2.3.1),
3. troubleshooting the subcomponents of a component (Table 5.2, Line 2.3.2.1), and
4. dealing with unsolved cases (Table 5.2, Line 2.3.2.3).

We can distinguish several ways of presenting milestone transitions to students, depending on the level of guidance required.

- In heavily guided practice, the computer should be capable of dictating the next milestone to be reached in the procedure. For example, "We have just determined that no fuel is flowing. We will examine each of the potential causes of this problem in turn."

- In more relaxed guidance, the student should be required to indicate the next milestone. For example, "We have just determined that no fuel is flowing. What do we do next in the fault tree?"

- Under even more relaxed guidance, the computer might simply note milestones as they occur and check to ensure that the student's actions are consistent with the schema. For example, "We have just determined that no fuel is flowing. You are checking the Starter Air Inlet Duct. Re-examine the fault tree to make sure that this is the next step."

Presenting the Structure of the Fault Tree

As with the interpretation schema, both a declarative and procedural interpretation of the tree itself is needed. The declarative presentation is perhaps best done graphically, using a display like that of Figure 5.3. As the procedure traverses the tree, the active elements can be highlighted in some way and/or subtrees can be displayed as a means of focusing attention (see Figure 5.6 for examples).

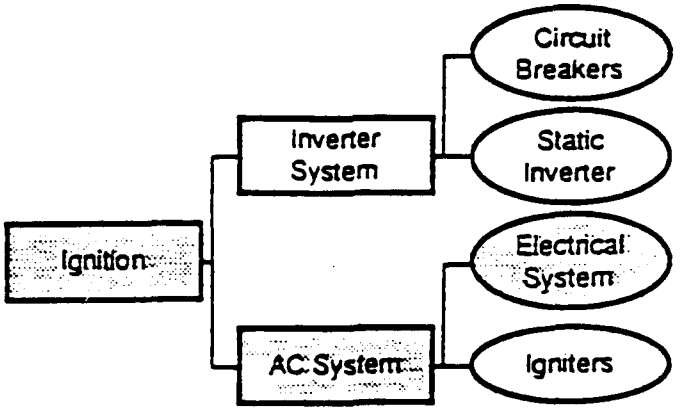


Figure 5.6 Use of subtree and highlighting to focus attention.

The procedural aspects of fault tree structure are defined as a list of functions in Section 5.3.2.2(3). A computer implementation of these functions should make their results available to students and should be able to query students about the structure of the tree. In particular, and referring to Section 5.3.2.2(3), the computer should have the means to:

- inform the student of the procedure needed to check the functionality of any component - for example, "To check for low fuel flow, use the procedure that checks fuel mist in the exhaust;"

- ask the student to designate or execute the procedure needed to check the functionality of a component - for example "Select the procedure for checking the overspeed generator;"

- inform the student of the procedure for repairing any faulted component - for example, "To repair the static inverter, use the procedure for replacing ~~the static~~ inverter;"

- ask the student to designate the procedure for repairing a faulted component - for example, "Select the procedure used to repair the disengaged IGNITION INVERTER circuit breaker."

These functions, can, like those listed in the previous section, be used in guided practice to ensure that the student learns how to properly manipulate the fault tree.

Presenting Procedures for Observation and Repair

Procedures for repair of faulted starting system components are presented in Table 5.1. Procedures for checking the functionality of components are listed in Table 5.3. As instructional objectives, both types of procedures are represented in some formalism such as GOMS (see Section 5.3.2.2(2)).

In a computer-based instructional system, these procedures need to be given a declarative, verbal description, such as plain-English paraphrase of the GOMS representation, and a procedural implementation in the qualitative and physical simulations. In this way students can be shown or asked how the procedure unfolds through interactions conducted in verbal terms, in terms of physical actions and observations, or in qualitative, conceptual terms.

The power of these separate presentation schemes is multiplied by joining them together in complete or partial presentations of the entire troubleshooting procedure. As the procedure is presented and practiced, the system or the student can focus on any of five critical aspects of the procedure:

- strategy - the status of the procedure with respect to the fault-tree interpretation schema;

- tactics - the status of the procedure with respect to the fault tree itself;

- stepwise descriptions - the elementary observation and repair procedures;

- conceptual aspects - the theoretical description (in the qualitative model) of the procedure;

- implementation - the physical actions and observations (in the physical simulation) that implement the procedure.

With the computer implementation suggested above, Halff views each of these aspects as being potentially available for presentation or practice at any point in the procedure.

(3) Problem Solving and Troubleshooting

Above, in Section 5.3.2.3, we discussed troubleshooting as a problem-solving activity as opposed to the fault-tree procedural approach described in Section 5.3.1.4. In Section 5.2 Halff suggests that the computational support needed for effective training of these problem-solving skills is a troubleshooting expert based on the problem-solving methods that students should master to solve particular problems.

Two such experts are available, namely, PROFILE (Towne, Johnson, and Corwin, 1983) and the Fuzzy Rule-Based Model of Hunt and Rouse (1984). The former is an almost pure information-theoretic approach that achieves not only efficient troubleshooting but also an impressive match to human troubleshooters. The latter, however, offers more face validity in that it incorporates some of the heuristics known to be used by human troubleshooters.

Halff does not say which of these models is more appropriate for training in this situation, but does say, however, that, whatever model is used, that model should operate within the context of the qualitative simulation described above. It should also be able to start in mid problem and, in particular, when the fault-tree procedure reaches an impasse. It should also be able to present or explain its choice of each troubleshooting action as problem-solving proceeds.

5.3.3.2 Instructional Methods

Instructional methods constitute the procedures for engaging the student in instructional interactions. Consistent with the view of instructional objectives described in Section 5.2, Halff proposes that some of these procedures address the acquisition of a mental model of the equipment. Some should address acquisition of procedures. Some should address the problem-solving skills needed for effective troubleshooting.

The foregoing section has provided us with a powerful set of tools for addressing these goals. To specify the instructional methods completely, we need to configure these tools and assemble them into a curriculum. The discussion above suggests that the curriculum for troubleshooting training should be organized around a few major types of activities that reflect advancing levels of knowledge and skill.

(1) System Behavior and Structure

The first level of understanding to be attained by students is that of a mental model of the device. Activities promoting such attainment include exercises with both the qualitative and physical simulations within a framework of the structural breakdown of the aircraft (Figure 5.3). Students at this stage will master fundamental qualitative reasoning tasks such as predicting the behavior of individual components and determining the implications of certain states of the equipment.

Exercises to be used in this phase of the curriculum are listed in Section 5.2.1.2. To quote from that section,

1. Physical and Conceptual Structure. Students are shown images of the physical equipment and asked to identify individual components, their function, and their immediate connections.

2. Causal Reasoning. Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode.

3. Functional Reasoning (a). Students are shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state.

4. Functional Reasoning (b). Students are shown the actual outputs and inputs to an element and asked to determine whether or not the element is faulted.

5. Physical and conceptual appearance. Students are asked to discriminate among component states on the basis of some physical depiction of those states.

The exact sequence of these exercises should be designed to reflect and convey the overall structure of the equipment. In the typical case, where the equipment can be hierarchically decomposed, the exercises can traverse this decomposition in a depth-first fashion so that students learn to reason about a subsystem immediately after learning to reason about each of its components.

These exercises should also be implemented with a view to whole-task training. Many if not all of them could be embedded in mini-troubleshooting problems in order to illustrate the application of qualitative reasoning to troubleshooting.

(2) System Function

A second phase of the curriculum should give students an understanding of how components function, or fail to function, together to meet the operational purposes of the aircraft. Put differently, students in this phase should learn:

- what the system and its components are supposed to do,
- how to make the system fulfill those functions, and
- how to determine when the system is not meeting its functions.

Activities at this level are organized around the functional breakdown of the aircraft, illustrated schematically in Figure 5.4. By making this structure evident in instruction, students should be able to induce the basic operations of the system and how its components are involved in those operations.

Activities addressing system function are operational in nature. Some exercises provide guided practice procedures such as in starting the engines and in elementary repair and observations of the type listed in Table 5.1 and Table 5.3. Practice should begin with the qualitative simulation alone, pass to a stage with joint qualitative and physical simulations, and end with the physical simulation alone. Each subphase should begin with demonstration of the procedure followed by practice.

The order of presentation of different procedures in this phase should conform to a depth-first traversal of the functional breakdown, and, more importantly to the subgoal structure of the procedures themselves. Specifically, the GOMS representation of each procedure breaks the procedure into small manageable subgoals that can be mastered individually. The curriculum should obviously take advantage of this feature of the representation.

A second type of activity in this phase should teach the students the tests or observations needed to check the functionality of each component of the aircraft. Several types of exercises can be used to effect such teaching.

- Students can participate in qualitative reasoning exercises that address the test procedures. For example, they might be asked to predict the behavior of the Right Fuel/Oxygen Indicator in aircraft that have normal or faulted static inverters.
- Students can practice observational procedures, such as checking fuel flow in the overspeed governor, in the context of testing the system's functionality. Exercises that place the student in an apprenticeship role can be used to implement this strategy.

- Students can be asked to both select and execute test or observational procedures. For example, where a beginning student would practice testing the AC bus by being informed that the test involved checking Pin N in the engine ignition and accessories disconnect plug, more advanced students would practice the same operation by simply being instructed to test the AC bus.

By the end of the second phase of training, students should have many of the component skills of fault-tree based troubleshooting procedures. In particular, they should know how to test any of the aircraft's components and subsystems for normal functioning, and they should have a good idea, from exposure to structures like that of Figure 5.4, of the structure of the fault trees to be used in troubleshooting.

(3) Troubleshooting Procedures

In the third phase, students should be introduced to fault-tree interpretation and to the fault trees that define the set of troubleshooting procedures to be learned. The activities that address these topics are described in some detail above in (2) Procedures Interpretation. Arranging these exercises into a curriculum is not a difficult task. The following guidelines seem reasonable in this respect.

- Every malfunction's fault tree should be covered first in a depth-first traversal of the tree and then in a random order.

- Initial exercises should provide explicit support for use of the fault-tree interpretation schema (see Section 5.3.1.2). This support can be withdrawn after practice with a few malfunctions.

- The first exercises with each malfunction should rely heavily on an explicit representation of the fault tree for the malfunction. This support should be faded with advanced practice.

- Exercises should begin first in the qualitative simulation. As students master the structure of the fault tree, use of the physical simulation can be phased in and support from the qualitative simulation can be faded.

- Documentation that is normally available in the field should be available on-line.

(4) Problem Solving

We have noted above the importance of teaching troubleshooting as a problem-solving activity. In Section 5.3.1.3, Halff suggested that problem-solving activities have a particular place whenever fault-tree methods arrive at an impasse (Line 2.3.2.3 of Table

5.2). Although students in the trouble-shooting procedure phase (Section 5.3.2.3) of the curriculum should be protected from such impasses, students ready for this problem-solving level should be presented with impasses as opportunities to practice problem solving.

Section 5.2 contains Halff's proposals for instruction in troubleshooting as problem solving. The following is a list of suggested exercises taken from that paper.

1. Troubleshooting. Students are provided with a conceptual simulation containing a single faulted component. At each point in the troubleshooting exercise, students would choose an action and exhibit the consequences of the action. The exercise could take many forms. For example, students might be prompted to select actions diagnostic of a particular fault or sets of faults. Other forms of troubleshooting practice can be found in Brown, Burton, and de Kleer (1982).

2. Reverse Troubleshooting. Students are told that a particular component is faulted. They are required to predict the results of certain observations based on this information. Causal reasoning patterns can be elicited or exhibited during the course of these exercises.

3. Case Studies. Students could be given real case studies of intractable troubleshooting problems. Computer support could be provided for collaborative problem solving and for peer and expert critiques of proposed solutions.

A typical troubleshooting curriculum might have the following lessons.

1. A set of reverse troubleshooting and troubleshooting problems that cover the major topological patterns found in the device. Each pattern would be addressed first by reverse troubleshooting exercises and then by troubleshooting exercises.

2. A set of reverse troubleshooting and troubleshooting problems that cover the equipment's mission-critical faults and their nearest neighbor. Students would first reverse troubleshoot each major fault and its neighbor and then troubleshoot the pair.

3. A repetition of Lesson 1 without reverse troubleshooting.

4. A repetition of Lesson 2 without reverse troubleshooting.

5. A mixture of Lessons 3 and 4.

Reverse troubleshooting in this curriculum plays the role of a cognitive support which is gradually faded from the curriculum.

Other cognitive supports (e.g. external hypothesis lists) should also be withdrawn in the last lesson.

What needs to be added to this description is:

- that these exercises should be introduced only in the context of an impasse in the fault-tree procedure, when, for example, all components under "Fuel" in Figure 5.2 function properly, but fuel is still not evident in the exhaust;
- that the exercises should be introduced only when the student has mastered the fault tree for the malfunction, and
- that the exercises should be conducted in the presence of a tutor working the problem under the same set of initial conditions as is the student.

5.3.4 Summary

The central contribution of Section 5.3 is the instructional analysis of troubleshooting procedures based on fault trees. The general approach suggested here to the teaching of fault-tree based troubleshooting can be summarized with the following points.

A fault-tree based troubleshooting procedure consists of a hierarchical fault tree, a general schema for interpreting the tree, and the elementary observation and repair procedures needed to implement the procedure. All three aspects of the procedure need to be mastered by students.

Training should begin, not with the procedure itself, but rather with instruction oriented to the structure and behavior of the equipment and its components. This initial instruction establishes the student's mental model and can be implemented using qualitative and physical simulations of the equipment.

Training on elementary observation and repair procedures should be introduced in conjunction with instruction on device functionality. This instruction should teach students how the equipment is used, how components operate together to achieve the functions of the equipment, and how to determine when the device or any of its components is not functioning properly.

Training on the use of fault trees should be based on guided practice. Initially the interpretation schema and the trees should be made explicit in the instruction. As students advance, this explicit support can be withdrawn.

When students are ready to master troubleshooting as problem solving, practice opportunities should be provided within the context of fault-tree procedures. In particular each problem-

solving exercise should begin by driving the fault tree procedure to an impasse in which a component malfunction cannot be traced to a malfunction in any of its subcomponents.

5.4 Computer-Based Maintenance Training

Halff thinks of an authoring system in terms of the authoring functions provided to the SME and the instructional products produced by the system. The instructional products are built up from transactions.

5.4.1 Authoring Functions

Authors should be provided with three editors. An equipment editor would be used to create simulated equipment. A procedure editor would be used to specify all of the operating and maintenance procedures to be taught. An instruction editor would be used to create exercises and lessons.

5.4.1.1 The Equipment Editor

The equipment editor should be nothing more than RAPIDS, at least in its first version. RAPIDS works and works well although there is no explicit distinction between physical and qualitative simulation, and there is no explicit requirement for a structural breakdown of the equipment. RAPIDS permits the development of these features.

5.4.1.2 The Procedure Editor

The author uses the procedure editor to specify all main operations of the equipment (see Figure 5.5 in Section 5.3), and, in the course of so doing, specify the maintenance procedures (including fault trees) that make up the course content. The procedure editor would normally be used in conjunction with simulation created by the equipment editor. The main stages of specifying the operations are as follows:

1. List the target operations of the equipment, for example, ground start of the T-38. Then, for each operation, execute the following steps.

2. Create a GOMS procedure (called the operating procedure) for executing the operation. This procedure would be created with a GOMS editor that worked in conjunction with a simulation of the equipment.

3. Specify all of the malfunctions that might occur in the course of the operation. These malfunctions would be specified in terms of observable states of the equipment simulation at particular points in the operation.

4. Specify all of the faults that could produce each of the malfunctions identified in Step 3.

(Steps 3 and 4 could be partially automated by running the operating procedure under all possible fault conditions of the device.

5. Write fault trees for each of the malfunctions identified in Step 3.

6. Use the GOMS editor to formulate the procedures needed to implement the fault trees created in Step 5. (See Section 5.3.2.2 and Tables 5.1 and 5.3 in Section 5.3).

The computational machinery needed to implement the procedure editor will consist of structure editors, text editors, and list editors.

A GOMS editor, in itself, would constitute a signal contribution to RAPIDS. The main components of the GOMS editor are:

- a recorder that would watch the SME operate the simulation and record the steps taken using the GOMS formalism;
- a control-structure editor that allows the SME to complete GOMS If, Decide, and Goto operators, and;
- a working-memory model that tracks the contents of working memory from step to step in the procedure.

5.4.1.3 The Instruction Editor

The instruction editor is the SME's interface to the instructional products, described next. Needed for this purpose is a facility for configuring transactions and a facility for constructing curricula.

The TRX editor permits the SME/ID to configure transactions with two sorts of data: (1) the material to be taught and (2) parameters that govern instruction. For example, Merrill's "parts" transaction must be informed of the system and its parts (the material to be taught) and the procedure used to teach these parts (the instructional parameters). An editor for configuring a transaction should allow the author to set both sets of parameters. It should also permit the author to systematically sample some parameters while constraining others.

The curriculum editor permits the author to assemble either author-specified or system-generated transactions into a curriculum; a bookkeeping system that represents the division of a canonical curriculum into phases and lessons, and permits the author to fill the slots in this curriculum.

5.4.2 **Transactions**

Curricula are to be constructed out of building blocks called transactions. (Transaction shells can be viewed as building-

block factories.) Here is a list of transactions derived from Sections 5.2 and 5.3.

System Behavior and Structure

1. **Physical and Conceptual Structure.** Students are shown images of the physical equipment and asked to identify individual components, their function, and their immediate connections.

2. **Causal Reasoning.** Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode.

3. **Functional Reasoning (a).** Students are shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state.

4. **Functional Reasoning (b).** Students are shown the actual outputs and inputs to an element and asked to determine whether or not the element is faulted.

5. **Physical and Conceptual Appearance.** Students are asked to discriminate among component states on the basis of some physical depiction of those states.

System Operation and Function

1. **System Operation.** Students practice all of the operating procedures in an unfaulted environment.

2. **Repair Procedures.** Students practice repair of particular faults using procedures such as those given in Table 1 of TTP.

3. **Observation Procedures.** Students practice the observations needed in the course of troubleshooting. See Table 3 of TTP for examples.

4. **Functionality Testing (a).** Students predict the results of functionality checks in normal and faulted conditions.

5. **Functionality Testing (b).** Students practice choosing a functionality test for particular subsystems and components.

6. **Functionality Testing (c).** Students practice testing the functionality of particular subsystems and components (i.e., a combination of 5 and 3 above).

Three general considerations drive the design of each of these transactions. First, all of these transactions should be taught

in an apprenticeship paradigm where the student is assisting a computer-based master technician. Thus, the practice called for in any transaction is provided in the context of a specific operational or maintenance task.

Second, procedure practice, such as that called for in Transactions 1-3, should be generated by formalizing epitome theory so that it applies to GOMS-modeled procedures. A single shell (which Halff calls the E-G Procedure Shell) might be designed to generate all of these transactions. Halff believes the result would look very much like VanLehn's step theory.

Third, all of these transactions are configurable as to the level of instructional support to be provided. In particular,

- the practice environment can include views from the qualitative model, and/or the physical model;
- guidance and feedback can be selectively provided;
- hidden or intermediate results (working memory in the case of GOMS procedures) can be explicitly shown;
- the control structure of the procedure can be shown and traced during execution.

Troubleshooting

1. **Fault Tree Interpretation.** Students practice the general interpretation of fault trees. The transaction shell for this transaction might well be a specialization of the E-G Procedure Shell mentioned above.

2. **Malfunction-Specific Troubleshooting Procedures.** Students practice troubleshooting the malfunctions identified in Step 3 of the process described above in the Procedure Editor section.

3. **Troubleshooting Problems.** Students practice troubleshooting faults not covered by any defined fault trees. A single exercise in this transaction should have the following steps:

   a. Drive the troubleshooting procedure to an impasse (Section 5.2, Table 5.2, Line 2.3.2.3).

   b. Use the IMTS Profile-based intelligent tutor to guide the student in completing the fault-isolation process.

These three transactions might be generated with successive modifications of the same shell. The configuration parameters of this shell include the following:

- Explicit support can be provided for the fault-tree interpretation process.
- The fault tree itself (appropriately depicted) can be made available to the student during the course of the exercise.
- Any of the configuration parameters identified above in the System Operation and Function section apply here as well.
- Faults requiring problem solving (i.e., not in any fault tree) can be excluded or included.
- Varying degrees of support, guidance, and feedback can be provided during the problem-solving process.

5.4.3 **Instructional Support**

The transactions described above ignore many of the aspects of instruction that contribute to coherence and motivation. Mechanisms are needed for orienting students to the curriculum, to individual lessons, and to special problems. Halff mentioned some of these functions briefly in Section 5.2.

5.4.4 **Instructional Products**

The instructional products of the authoring system are computer-based maintenance-training curricula. These curricula consist mainly of sequences of the transactions described above. In Sections 5.2 and 5.3, Halff describes the main constraints on these curricula, but does not describe how the authoring system can turn this list of constraints into a complete curriculum.

At this point, only three approaches suggest themselves to Halff, none of them ideal.

First, SMEs could construct curricula by hand. Each exercise would be constructed by choosing a transaction shell, selecting the subject matter needed to configure the shell, and setting the instructional parameters of the shell. This method could be supported with on-line documentation and bookkeeping to promote systematic and effective curriculum construction, but it would still impose an incredible burden on authors of new courseware.

A second approach would use the sequential constraints defined in Sections 5.2 and 5.3 to generate exhaustive curricula containing every possible transaction. SMEs could then select transactions from the complete set to form a realistic curricula. Since the exhaustive curriculum would be generated from explicit constraints, the system could warn authors of any constraints violated in the curricula that they construct.

A third approach is to develop a programming language that would provide access to transaction shells. This language, properly designed, could be used to define curricula in much the same way

as conventional languages do now (including that provided with RAPIDS). It could also be used to design individualized instruction (as opposed to a fixed curriculum) and could be used to implement the first two approaches described above.

The AIDA specification might well call for the development of all three of these approaches.

5.4.5 **Other Considerations**

What is described above is far less than a complete authoring system. Halff has limited himself to those central components that implement the concepts in Sections 5.2 and 5.3, two components being critical.

- On-line help is needed to guide instructional development and provide assistance on system functions.

- A library is needed for retention and reuse of any and all products of the authoring system. These products may range from individual component models to complete curriculum. Obviously, an appropriate database should be used to organize the library.

General design considerations include:

- Authors should be able to work in parallel or multiple aspects of a course whenever possible. The system should propagate constraints from one aspect of the course to others and should make available only those options that apply at any particular time.

- Authors should be able to try out any product or subproduct of the authoring process, including the simulation, procedures, and transactions.

5.5 Analysis of Maintenance Tasks*

5.5.1 Introduction

The first step in the design of any instruction is a task analysis to determine what should be taught. From the cognitive science information processing approach, it is argued that a behavioral analysis is not sufficient. A cognitive analysis needs to be performed because education and training should take into account the cognitive processes involved in learning and performance, not just the objective behaviors required (See Glazer & Bassock; 1989, as well as several chapters in Psotka, Massey, & Mutter, 1988 for a recent discussion of this issue).

In Section 5.2 Halff identified three types of cognitive structures important to the maintenance enterprise: the execution of procedures, a mental model of the equipment, and fault isolation skills. Thus, an adequate cognitive task analysis should identify the information and skills that must be imparted to the student to support the acquisition of these cognitive structures. Only the procedures and mental models are addressed in this section.

The walk-through of generating with AIDA, training materials on the T-38A aircraft, is partly based on two cognitive task analyses performed by David Kieras of the University of Michigan. The first, presented in Kieras (1988b), centered on the issue of what mental model should be taught concerning the engine ignition system of the T38; it was used in constructing the example training materials in the RAPIDS II Authoring Manual. The second, generated by a subcontract to this effort, is a cognitive task analysis of the troubleshooting procedures for the fault of "no start" in the T38 engine.

While the current walk-through is based in part on Kieras's cognitive task analysis, little attention has been paid to date to specifying the nature of the task analysis in the AIDA system. The primary focus of this paper is to explore the task analysis conducted by Kieras as a basis for specifying the task analysis requirements in an AIDA designed for maintenance training. The questions to be considered include:

- What is the nature of the cognitive task analysis?
- How detailed does the analysis need to be?
- How should that task analysis be represented in AIDA?
- How do you map the representation onto the instructional materials?
- What kinds of aids and/or guidance could be provided to an instructional designer, who might also a be subject matter

* This section was prepared by Martha C. Polson, Univ. of Colorado, and David Kieras, Univ. of Michigan.

expert (SME), but a novice instructional designer to perform the task analysis?

This section does not attempt to provide complete or final answers to the above questions, but primarily strives to spell out the issues that need to be addressed and to list some of the relevant literature.

5.5.2 **Cognitive Analysis of Procedures**

The analysis of the troubleshooting procedures done by Kieras is a particular type of analysis known as a GOMS (Goals, Operators, Methods, and Selection Rules) analysis, which derives from the Cognitive Complexity Theory (CCT) of Kieras and Polson, (Bovair, Kieras, & Polson, 1990) and has as its intellectual predecessor the work of Card, Moran and Newell, (Card et al., 1983). This approach entails analyzing the tasks to be accomplished into a meaningful series of goals and subgoals. Each goal to be accomplished is recursively broken into a series of subgoals until a level is reached in which accomplishing the subgoal can be achieved by either a primitive level motor or a mental act. Such a simple act for the T-38 start system would be to press the left start button or apply the shorting stick.

Goals represent a person's intention to perform a task, subtask, or single cognitive or physical operation. Goals are organized into structures of interrelated goals that sequence methods and operations. An example goal from troubleshooting the engine would be to determine if the ignition system is functioning correctly.

Operations characterize elementary physical actions (e.g., pressing a button, setting a switch, or attaching a probe) and cognitive or mental operations (e.g., perceptual operations, retrieving an item from memory, or reading a voltage and storing it in working memory). The most primitive mental operations are actions such as receiving perceptual information, making a basic decision, depositing facts from working memory into long term memory, retrieving facts from long term memory and activating them in working memory, forming a goal, etc.

Methods generate sequences of operations that accomplish specific goals or subgoals. The goal structure of a method characterizes its internal organization and control structure. The GOMS model assumes that execution of a task or procedure involves decomposition of the task into a series of subtasks. A skilled person executing a procedure has effective methods for each subtask. A novice may have less efficient methods. Accomplishing a task involves executing the series of specialized methods that perform each subtask. There are several kinds of methods. High-level methods decompose the initial task into a sequence of subtasks. Intermediate-level methods describe the sequence of functions necessary to complete a subtask. Low-level

methods generate the actual user actions necessary to perform a function.

A person's knowledge of how to do a complex task is a mixture of task-specific information - the high-level methods - and system-specific knowledge - the low-level methods. A high-level method for troubleshooting the T-38 engine would be:

- check out the starting operations of the engine.

Intermediate methods which are part of the high-level method for checking out the symptom of no-start would include:

- check for bad ignition
- check for fuel flow problem
- check for defective starting system
- check for altitude-limitation problem

Low level methods for the intermediate methods include:

For checking for bad ignition:

- apply shorting stick to after burner plug
- check the ENGINE IGNITION, R AUTOSYN INST & IGNITION INVERTER circuit breakers for proper engagement

For fuel flow problem:

- check fuel system circuit breakers for proper engagement

Selection rules determine which method to select. In an expert, selection rules are compiled pieces of problem-solving knowledge. The selection rule must state the appropriate context for using any given method. If there is more than one method, the rule must state when each method is appropriate.

In summary, the GOMS model characterizes the user's knowledge as a collection of hierarchically organized methods and associated goal structures that sequence methods and operations. The knowledge captured in the GOMS representation describes both general knowledge of how the task is to be decomposed and specific information on how to execute the methods required to complete the task.

One of the greatest advantages of this approach is that Kieras has prepared a detailed guide for doing task analysis of procedures using the GOMS methodology (Kieras, 1988a). He has also defined a language call (NGOMSL) or "Natural" GOMS Language which is relatively easy to read and write. Kieras' guide also includes procedures for doing a GOMS analysis by using a breadth-first expansion of methods rather than trying to describe goal structures directly.

5.5.3 **Mental Models Analysis**

Halff (1990) summarized the importance, for maintenance training, of imparting correct and adequate mental models of the equipment. Kieras (1988b, 1990) pointed out that the most accurate way of determining the mental model to be taught would be to do a complete cognitive simulation. However, realizing that this is not always a feasible approach, Kieras (1988b) spelled out some heuristics that could be used to determine the mental model that should be taught in lieu of a complete simulation. The heuristics are:

- relevance to task goals
- accessibility to use
- critical procedures and inference strategies

All of these heuristics involve doing an analysis equivalent to a GOMS analysis of the task at hand. In addition two other hierarchical cognitive analyses are required, an explanation hierarchy and a hierarchical decomposition of the device structure and mechanisms.

The first heuristic, relevance to task goals, states that explanations should be given only if they are relevant to a task goal. To carry out this heuristic, an explanation hierarchy is constructed. The first pass at what goes into this hierarchy can be what is in the existing documentation. The goals of the GOMS analysis are then mapped to the explanation hierarchy, which will reveal any missing explanatory information as well as any extraneous material which need not be taught. Constructing the explanation hierarchy is not really extraneous work since this material is needed for the instructional material. For instance this is the material that goes into the message windows in the RAPIDS system.

The second heuristic, accessibility to use, implies that the device illustration or simulation which is presented to the technician should not contain parts which he cannot access.

Again this involves mapping the GOMS analysis, but onto the device description, rather than the explanation hierarchy.

The third heuristic says that the GOMS analysis should be examined for procedures that will be difficult to learn due to what appears to be arbitrary content. These procedures should then be analyzed to determine what inferences would need to be made in order for the content to appear logical rather than arbitrary. The information necessary to make those inferences should then be made explicit in the training materials. This information will need to be included either in the explanation hierarchy or the device description.

5.5.4 **Level of Detail of the Task Analysis**

Kieras (Kieras, 1990) as well as Anderson (Anderson, Boyle, Corbett, & Lewis, 1990) has advocated doing a complete cognitive simulation of a given task which is based on a cognitive analysis of the task in order to determine the content of instructional materials and training procedures. The advantage of a simulation is that it insures that the analysis is complete. Also, as Kieras and Polson (Bovair, Kieras & Polson, 1990) have shown, a tremendous amount of information about the task at hand can be gained if the analysis is completed down to the level of simple operations or operators for most aspects of the task. The information that can be derived from the simulation includes the time to learn the task, the amount of transfer of training from one procedure to another, and the execution time for various procedures or methods. The disadvantage is that a complete cognitive simulation requires a tremendous amount of effort to implement, even after the cognitive analysis of the content of the instruction is complete. However, as can be seen from the GOMS analysis of Kieras, the use of the GOMS method for cognitive analysis of procedures does not require that it be followed through by a complete simulation or that all tasks be analyzed to the level of simple operators.

How low the level of analysis needs to be for the procedures for any given instructional package, will be determined in large part by the level of expertise of the trainees. For instance, for the problem of No-Start with the Probable Cause of no ignition or poor ignition, the first step is to check ignitor plugs for firing and proper spark rate. This step is followed by a note that the proper spark rate is 3 sparks in 2 seconds (See Figure 5.3). Presumably this is as low as the analysis needs to go. In terms of a computer-based instructional system, this detail could be represented by clicking on a designated ignitor plug icon handle (handles are mouse sensitive areas) which will give its status. If the status had been set to bad, then the simulation would continue with the procedure (the next step, if the ignitor plugs do not fire, is to check the static inverter). The actual motor and perceptual operations necessary in checking the spark rates would not have to be explicitly laid out. However, for a novice technician, some of the steps in the T-38 troubleshooting manual, such as "remove the engine" do seem rather high level and may need to be broken down into subtasks. Anderson (Anderson, Boyle, Farrell, & Reiser, 1984) refers to this subtasking as adjusting the grain level of the instruction.

As a way of decreasing the workload of authoring the simulation and/or doing the GOMS analysis for a given domain, a library of generic low-level procedures such as testing ignitor plugs (as well as their corresponding simulations) could be provided in an AIDA configured for that domain. In fact this library could include a set of separate modules that are given as screening tests to insure that these low-level methods or methods which occur in many different troubleshooting situations, such as

"remove the engine" are learned before entering simulations which are higher level or aimed at specific problems. A problem for a generic system, as opposed to a system written explicitly for a domain such as electronic maintenance or airplane maintenance, is knowing what skills and knowledge can be assumed. If the domain is known, there is probably a reasonably finite set of testing skills, mechanical procedures, etc. that are known to be required to perform the task. For instance if the student is said to be at a particular skill level in a particular field, is there a list of basic procedures that the student can be expected to know and which would not have to represented in detail in a particular domain?

5.5.5 **Representing The Task Analysis**

In the CCT approach of Kieras and Polson, a simple production system is used to implement the results of a NGOMSL analysis into a working simulation. The device knowledge necessary to carry out the simulation is represented in a Generalized Transition Network (GTN)2 (Kieras & Polson, 1985). However a number of representation schemes are possible. A scheme used by Anderson in his PUPS system is a candidate representation that is probably compatible with the Transaction Shell representation discussed by Merrill (Jones, Li, & Merrill, 1990). Anderson's PUPS (Penultimate Production Systems) theory holds that procedures are acquired by compiling declarative knowledge (Anderson et. al., 1990). The declarative knowledge necessary for compiling the procedures which model the task performance is represented in schema-based structures called PUPS structures. These schema include slots for the function of the entity being represented by the schema, a form slot for the physical appearance of the entity, and a precondition slot which state the preconditions necessary for the function to be achieved (Anderson et. al., 1990). In compiling the productions which are the basis of procedural knowledge, the function slot maps to the goal to be achieved which will require knowledge of the entity represented; the preconditions slot maps onto the condition of the condition-action pair in a production. The form slot in the PUPS tutors holds the form of the current action to be carried out such as a particular LISP function. A similar scheme could be used for representing the GOMS analysis. Merrill has proposed an activity frame that has paths or sequences of actions. This frame could also have slots for the function, the operators, and the outcome. The values for these slots could probably be automatically generated from a NGOMSL analysis just as it is technically feasible to generate a running production rule-based simulation from a NGOMSL analysis.

The explanation hierarchy can be represented in numerous different ways. It appears that the representation scheme already proposed by Merrill for AIDA (Jones et. al., in press) would be adequate to represent the explanation hierarchy. The device knowledge will ultimately be represented in the graphical simulation. The initial representation may be a hierarchical

listing of the names of the device components or perhaps a block diagram, which can serve as a guide for constructing the sketch which, in turn, will guide the construction of the graphical simulation.

5.5.6 **Mapping the Content of GOMS and Mental Model Analysis To the Device Simulation**

Following Kieras's approach will yield three hierarchically arranged representations. The GOMS analysis will spell out the steps to be followed in carrying out procedures for operating, calibrating, troubleshooting, or repairing the equipment starting with the highest level goals and methods. These are succesively decomposed to lower-level subgoals and methods. The GOMS analysis will also identify any device components that need to be included in the representation of the device structure as well as the declarative knowledge that needs to be conveyed about them -- function, location, name, etc. The explanation hierarchy will contain the causal and declarative knowledge necessary to execute the procedures, support inferences necessary for constructing a mental model of the equipment, and define the attributes and rules of objects, etc.

The device simulation in a system such as RAPIDS contains a graphic representation of the device structure and qualitative simulations of its functioning. Authoring in the RAPIDS II simulation starts with a temporary sketch which is derived from the prior cognitive analysis, particularly the mental model analysis which entails inter-relating the GOMS analysis, the explanation hierarchy and the hierarchical device structure decomposition. However, the construction of the simulation is done in bottom-up fashion starting with the lowest level of the device hierarchy. The lowest level objects are the bottom items in the device structure analysis. These correspond to the objects manipulated by the lowest level operators in the GOMS model. For this reason, it is not feasible to develop the simulation and do the GOMS analysis and explanation hierarchy in parallel, which might be tempting to the novice instructional designer, who wants to get on with "real" work. The analyses have to be complete before the construction of the simulation can begin.

The behavior of the objects are defined by attribute handles and rules. These aspects of the simulation are drawn from the explanation hierarchy. Once the basic simulation is complete, procedures which are carried out on the device are authored by carrying out a sequence of actions which correspond to actions spelled out to accomplish the goals in the GOMS analysis. The individual actions correspond to the operators. What is missing from the simulation representation is any indication of the function or purpose, i.e. goals, of the procedure which have to be represented in the dialogue windows.

5.5.7 **Aids for Doing a Cognitive Analysis**

As mentioned in Polson & Polson (1990) it should not be difficult to implement a shell which can guide a novice in doing a GOMS analysis of a particular task using either the documentation at hand or the knowledge of a subject matter expert. The shell can be based on the previous work of Kieras (Kieras, 1988a) who has invested a large amount of time in writing a manual on how to do GOMS analysis and in developing an English-like language for representing the analysis. Included in the guide are many rules of thumb which could be implemented in a knowledge-based shell to give guidance to the SME or instructional designer. For instance, Kieras recommends that a given method contain no more than five steps. If there is more than that, some may need to be grouped into a higher level method. There is also guidance on creating generic methods to represent methods which occur often in slightly different context. For instance, rather than a method for checking each specific circuit breaker, there would be a check circuit breaker method, which has as a variable which circuit breaker to check. This variable information is held in working memory.

This shell could do much of the bookkeeping necessary for a GOMS analysis such as creating a list of methods and information identified by the methods that need either already to be known or taught, such as their location, etc. A more sophisticated shell could automatically map the results of the analysis into the knowledge representation system. A less sophisticated system would create a paper guide for what should be hand entered into the representation system. Similar shells could also be created for the explanation hierarchy and the device structure and function knowledge. Explicit guidelines for doing such analyses are not yet available, however.

How difficult a given task analysis will be, and the type of guidance that will be needed, will depend to a large part on the nature of the documentation. In some T.O.s the steps in carrying out a procedure are spelled out in excruciating detail. However, the T.O. lacks any hint of a goal structure or any supporting material for creating a mental model which could guide the performance of the task. The problem in doing a cognitive analysis of the task would be providing the information in the form of the higher level goals and methods and the explanation hierarchy. It could be noted that many T.O.s could benefit from the heuristic of presenting a general method and noting with variables to which segments of the equipment that it would apply.

Other T.O.s, notably Technical Order 1T-38A-2-6-2 for engine conditioning of the T-38A aircraft, go to the other extreme in that single steps are at the level of "remove the engine". However, the general goal structure is represented in the "trouble" and "probable cause" headers which are always visible.

Conclusions

Polson recommends that the task analysis approach developed by Kieras and his colleagues be adopted for the task analysis module of AIDA (KARS). This includes a GOMS analysis for the procedural aspects of the task and performing a mental model analysis by way of developing an explanation hierarchy and a decomposition of device structure and function and relating them to the GOMS analysis. Developing shells to aid in the cognitive task analysis is technically feasible. However, a great deal of care will need to be taken to be sure that the shells are implemented in such a way that the instructional designer perceives them as an aid, not as a hindrance or an extraneous useless requirement.

5.6 XAIDA Functions Applied to the T-38 Engine Starting System*

Reigeluth visualized 14 steps or functions XAIDA would perform to guide the instructional designer.

5.6.1 Confirm Sequencing Strategy

Input: An expert (SME) in the maintenance of the T-38 engine starting system.

Process: XAIDA prompts the instructional designer to interview the SME to find out whether the task can be easily proceduralized.

Output: Confirmation that the task is procedural; Selection of the template for "procedure" task analysis.

5.6.2 Identify a "Just Simple Enough" Class of Cases

Input: Job situation; Several task experts; Several marginal target learners (lowest entering ability) or an instructor very familiar with their entry-level abilities; Information about the environment and learners to decide on the optimal size of a module.

Process:

- XAIDA prompts the designer to have the SME think about what makes some "engine starting system maintenance" cases easier than others, and then to think of the **simplest class** of cases the SME ever performed.

- It then prompts the designer to develop (with an experienced instructor) an estimate as to **how many hours** of intensive learning time it would take a target learner to learn to perform that simplest class of cases as an expert would perform it (including time to develop all necessary mental models).

- XAIDA prompts the designer to decide **how long** a module of a training course should be (approx. 3-10 hours of learning time). If the amount of time required is too long or too short for a single module, XAIDA prompts the designer and task expert to **further simplify or expand** the simplest class of cases.

- Then XAIDA prompts the designer and task expert to list the **conditions** that make the simplest class simpler than the most complex class of cases.

Output: Identification of simplest class of cases and its simplifying conditions:

* This section was prepared by Charles M. Reigeluth, Indiana University.

- Simplest class: Restart left engine when right engine is running.

- Simplifying conditions: Right engine already running (no need for power connections), Functional testing (no exterior or interior inspections needed), No engine testing needed, No troubleshooting needed, No emergency procedures needed, ...

5.6.3 **Identify Progressively More Complex Classes of Cases** (Can be done after 4)

Input: Simplifying conditions (output of Function 2).

Process: Rank order the simplifying conditions on the basis of how important and representative of the whole task its corresponding class of cases is.

Output: A simple-to-complex sequence of classes of cases for the task:

- Start right engine when plane is on ground (requires power hook-ups).

- Restart engine after maintenance (requires interior and exterior inspections).

- *Engine problems--diagnosis [Section IV]:* operating limits, instrument tolerance, ... (requires diagnosis procedures).

- Engine maintenance testing [Section V] (requires operating tests and inspections).

- Engine troubleshooting [Section VI]: No start, slow start, RPM hang-up, hot start (requires troubleshooting procedures).

- Engine problems--emergencies: fire, overtemperature, overspeed, smoke/fumes, oil system, generator, hydraulic system, compressor stall, engine flameout (requires emergency procedures).

5.6.4 **Conduct Task Analysis on Each Class of Cases**

Input: Output of Functions 2 and 3; Job situation; Several task experts; Several marginal target learners (lowest entering ability) or an instructor very familiar with their entry-level abilities.

Process: For each class of cases, XAIDA presents templates for the designer to fill in while interviewing the SME.

- There is a **device template** which prompts the designer to input a diagram of each device (or each variation of a

device) operated upon in performance of the task for the simplest class of cases.

- Then XAIDA prompts the designer to identify (from the SME) the **alternative procedures** that an expert would use to perform the simplest class of cases, and to input a label for each. (In some situations there may be only one alternative.)

- Then XAIDA sets up a **procedure template** for each alternative procedure and prompts the designer to conduct a procedural task analysis with the SME) to fill in the template. This analysis identifies all **steps** (at description entry level) in each alternative procedure, in the order in which they need to be performed, along with the **objects** (or parts of devices) that are acted upon in each step of the procedure and the **tools** that are used in performing each step.

- Using the results of the procedure analysis, XAIDA generates **kinds taxonomies** and **parts taxonomies** for the objects and tools, using appropriate templates. The SME is asked to modify and expand them as appropriate. It also develops a graphic **physical model** of each device, by asking the SME to label each of the parts (objects) on the diagrams entered earlier.

- There is a **functional model template** (referred to by Henry Halff as a conceptual model) which prompts the SME to create a schematic representation of how each device works. It can also be applied to tools when appropriate.

- XAIDA prompts the designer on how to **confirm** the results of the analysis with other SMEs and designer observation of the task.

Output: A procedural model for each alternative procedure, kinds and parts taxonomies for all objects and tools, and a physical model and a functional model for each device (and each tool as appropriate), all validated by several SMEs.

For simplest class: Restart left engine when right engine is running:

- Procedural model [2-6]: 1) Clear danger areas; 2) Signal ground crewman to apply external air; 3) Push engine start button momentarily; 4) Advance throttle to idle at 14% min. RPM; 5) Check ...; 6) ...; 7) ....

- Taxonomies: Certain kinds of instruments and controls in cockpit (only those that will be used during the procedure).

- Physical model for each device: Cut-away drawing of parts of airplane the learner will be using in the simplest class of cases.

- Functional model for each device: Schematic drawing (preferably dynamic) of parts of airplane the learner will be using in the simplest class of cases.

5.6.5 **Design the Sequence of Major Content**

Input: Output of Function 4.

Process: XAIDA executes an algorithm which prepares an outline of the sequence for teaching each class of cases. For a given class of cases, one alternative procedure is picked for one kind of device, and:

1. The **functional model** for that device comes first.

2. The **physical model** for that device (including all of its parts or objects) comes next and is related to the functional model.

3. The **parts taxonomy** for the device comes next (as a synthesizer).

4. The **procedural model** comes next, with its entry-level steps sequenced in the order in which they are performed on the job, and **each tool** being listed just before it is needed in the procedure.

5. The remaining **taxonomies** are presented as synthesizers.

The same kind of sequence is outlined for each additional device and procedure for this class of cases, and for each subsequent class of cases.

The designer and SME can modify the outline as they see fit.

Output: An outline of the sequence for all major content to be taught for all classes of cases.

For simplest class of cases: Since there is only one functional model, one physical model, one taxonomy, and one procedural model, the sequence is as outlined under process above.

5.6.6 **Analyze Supporting Content** (Can be done after Function 3 or 4)

Input: Output of Function 3 or 4; Several SMEs; Several marginal target learners or an instructor very familiar with their entry-level abilities.

Process: For each class of cases, XAIDA prompts the designer and SME to fill out slots for supporting content. The supporting content includes primarily principles, attitudes, information, and prerequisite concepts and discriminations. The templates are based on the Elaboration Theory's content analysis procedures, including Gagne's hierarchical analysis, and on Merrill's recent work. The designer can also modify any template for any given step.

Output: List all supporting content to be taught with each step of the procedure. This includes learning prerequisites, relevant principles and concepts, related attitudes and values, and useful information.

5.6.7 **Design a Content Sequence for Each Module**

Input: Output of Function 5.

Process:

- Based on the earlier decision about how long a module should be, XAIDA prompts the designer to allocate major content and its related supporting content to modules, using estimates from an experienced instructor as to how long it will take to teach the content.

- Then XAIDA applies rules (asking questions of the designer or task expert when necessary) to generate an outline for a within-module sequence of content for each module.

The designer can modify the content allocation and module sequences as needed.

Output: A clustering of all major and supporting content into instructional modules; An outline for the sequence of all content within each module, including simulations that provide integrated demonstrations or practice for the whole task or part-tasks.

5.6.8 **Classify Micro Content**

Input: Output of Function 5; Several SMEs.

Process:

- XAIDA presents default classifications for type of learning (based on a few simple decision rules) for each piece of content, and asks for confirmation from the SME (with the help of the designer), based on that person's determination of post-instructional requirements. It then requests confirmation from a second SME, time permitting.

Output: Classification of type of learning for each piece of content in each module.

5.6.9 **Decide on a Strategy for Each Cluster of Content**

Input: Output of Functions 6 and 7.

Process: For each module, decide what will be taught by programmed tutorial, by drill and practice, and by simulation. Some content may be taught by several strategies (e.g., the procedural model may be taught via generality-demonstration-practice-feedback in a tutorial [low fidelity of representation], followed by additional demonstration-practice-feedback in a simulation). Revise the sequence of content for the module, as appropriate.

Output: Allocation of content to strategies, and revised sequence of content.

5.6.10 **Select Appropriate Template for Tactics for Each Piece or Cluster of Content for Each Strategy**

Input: Output of Function 8.

Process: XAIDA uses a simple matching algorithm based on type of learning and type of strategy to select a "lean" template for each piece or cluster of content.

Output: Allocation of a lean template to every piece or cluster of content that has been selected for instruction.

5.6.11 **Analyze Micro Content**

Input: Output of Functions 5, 7, and 9; Several SMEs; instructors familiar with the target learners.

Process:

- For each piece or cluster of content, XAIDA prompts the designer to elicit learning difficulty levels (on, say, a scale of 1-5) from an instructor familiar with the target learners. It also requests confirmation from a second instructor, time permitting. (Later, during the formative evaluation, it will test and revise those estimates.)

- For each skill, XAIDA prompts the designer to elicit dimensions of divergence and the important variations for each dimension from the SME.

- For each simulation, XAIDA prompts the designer to elicit a scenario and a causal model (qualitative or quantitative) to govern the computer's actions in the simulation.

- XAIDA also requests confirmation from a second SME, time permitting.

5.6.12 **Modify Micro Templates**

Input: Outputs of Functions 9 and 10.

Process: XAIDA uses production rules to expand existing lean templates (from Function 9), including slots for examples and practice for each variation of each dimension of divergence; and it uses decision rules to replace existing templates (from Function 9) with more elaborate templates. It also uses matching algorithms to select appropriate templates for the proctor's guide for each template in the computer-based instruction.

Output: Expansions of the templates from Function 9, to include slots for tactics appropriate for higher levels of difficulty, and slots for examples and practice for each variation of each dimension of divergence; a template for a proctor's guide for each module.

5.6.13 **Develop Instructional System**

Input: The output of Functions 6 and 11; Several task experts.

Process: XAIDA prompts the SME, under the watchful eye and clarifications of the designer, to fill in all templates with words and graphics, modifying any templates as they see fit. XAIDA automatically programs and compiles the CBI and automatically prints out the proctor's guide after each has been reviewed and confirmed by other SMEs.

Output: A complete instructional system, including computer-based instruction, tests, and proctor's guide.

5.6.14 **Formatively Evaluate and Revise**

Input: The output of Function 12; Several learners from the target population; Several task experts.

Process: XAIDA collects data from learners from the target population as they proceed through the instruction. It identifies weak points in the instruction, and proposes solutions for approval or modification by the designer and task experts. All approved solutions are automatically made by XAIDA, the program is recompiled, and the proctor's manual is reprinted.

Output: A revised instructional system that is proven effective.

SECTION 6. A GENERAL THEORY OF INSTRUCTIONAL DESIGN*

6.1 Introduction

This section describes Merrill's Second Generation Instructional Design Theory. It combines the details of the instructional design process with a walkthrough of an example based on maintenance troubleshooting of the starting process for the T-38 dual engine jet aircraft.

The balance of this section is divided into four sections:

6.2 Knowledge analysis
6.3 Transaction authoring
6.4 Strategy analysis
6.5 Instructional delivery

These steps comprise the course development process. While presented in order, it should be recognized that the steps are actually performed iteratively, in a spiral fashion, with important linkages and sharing of data between the steps.

6.2 Knowledge Analysis

Knowledge analysis is the step whereby knowledge of the domain to be instructed (in this case, aircraft maintenance procedures) is elicited from subject matter experts and represented in a domain knowledge base which may be used by all other steps and tools in the course development process. For this purpose Merrill developed a domain-independent knowledge representation model (Figure 6.1).

6.2.1 The Knowledge Representation Model

Knowledge is represented by objects Merrill calls frames; each frame has an internal structure, and external links to other frames. These external links are termed elaborations of the frame. A set of elaborated frames linked together, containing all the knowledge required for instruction leading to acquisition of an integrated human performance, or enterprise, is called an elaborated frame network.

There are three kinds of frames:

- entities, corresponding to some thing, for example a device, object, person, creature, place, or symbol;
- activities, groups of related actions to be performed by the learner; and

* This section was prepared by M. David Merrill, Utah State University.

- processes, groups of related actions entirely external to the learner.

There are four kinds of elaborations. These are:

- attributes, which represent characteristics of a frame;
- components, which represent constituents of a frame. For an entity, the components would be parts of the entity; for an activity, steps; and for a process, events and causes;
- abstractions, which correspond to a "kinds-of" class/sub-class hierarchy into which the frame may be classified;
- associations, links to other frames in the network.

The network structure of the knowledge representation allows information to move through the structure, so that data contained in one part of the net affects the data stored elsewhere. Two principal means by which this occurs are:

- inheritance, in which attributes of a class or super-class in an abstraction hierarchy are passed to a sub-class or instance;
- propagation, in which the contents of a frame influence the contents of another frame connected to it via an association link.

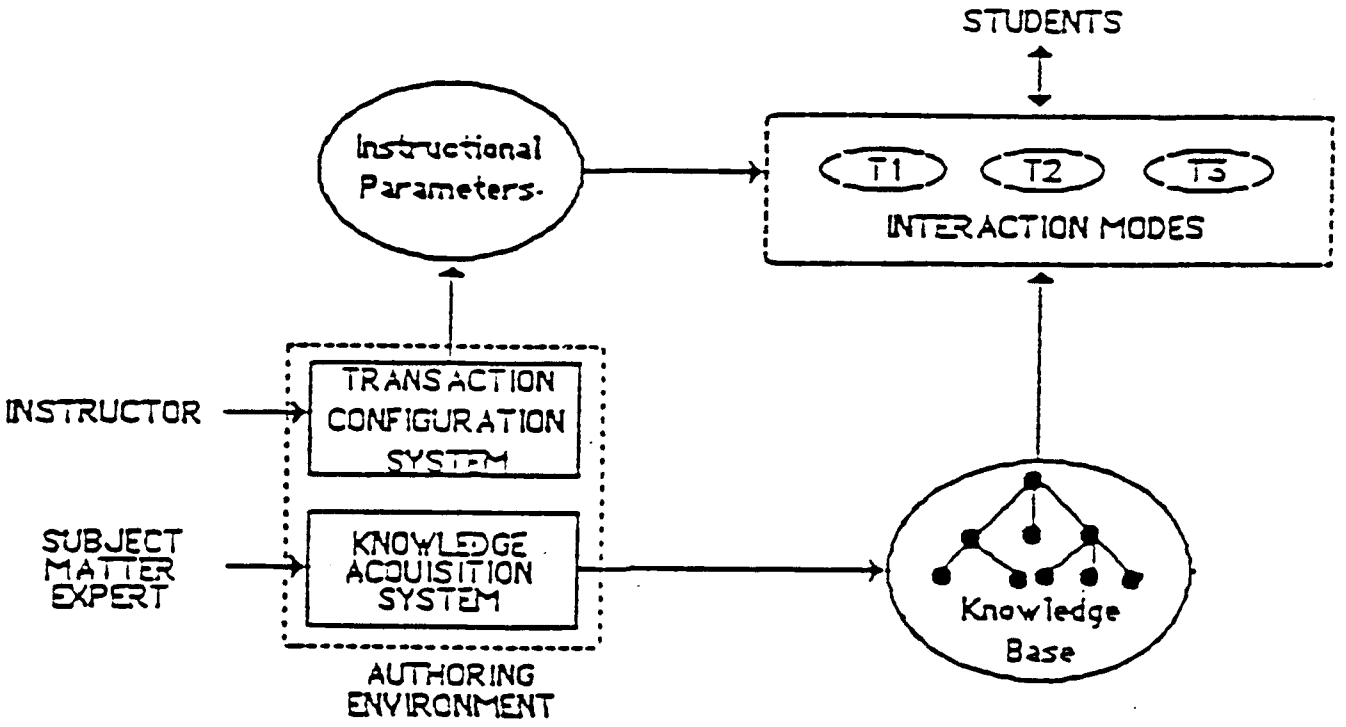


Figure 6.1 Components of an instructional transaction shell.

The knowledge representation model focuses on identifying aspects of knowledge which may serve as a basis for making instructional decisions.

6.2.1.1 Entity Frames

Entities are things in the real or imagined world including objects (natural objects and manufactured devices), creatures (animals and persons), places (natural and constructed), and symbols.

Examples of entities include the Eiffel Tower, a tractor, George Washington, 1.

We assume that there can be no instruction without at least one entity involved. The only enterprise that may be performed with only an entity is denoting.

6.2.1.2 Activity Frames

An activity is some group of actions performed, or which could be performed, by the learner.

Examples of activities include operating a device, using a formula to perform a calculation, and participating in a social interaction, such as group decision-making.

6.2.1.3 Process Frames

A process is some group of actions outside the learner including physical and social events in the real or imagined world. Examples of processes include the functioning of a device, the transmission of a disease, decertification, cell replication, planetary motion, group decision-making, and evolution.

The distinction between an activity and a process has to do with the role of the learner. If the learner would be, or could be an actor, then the actions are analyzed as an activity, with the learner's potential role central to the analysis and instruction. If, on the other hand, the learner could not be an actor and the actions occur entirely external to the learner, then the actions are analyzed as a process. For example, the procedure to service a machine, which will be performed by the learner, is an activity; replicating a cell in biology is a process.

6.2.2 **Relations**

In addition to the frame, the other fundamental structure in ID2 is the relation.

Relations are structures that link and attach meaning to a set of frames. Each frame links to the relation, and from the relation

to other frames in the set, in a manner specified by the relation. The semantics of the particular relation give meaning to the individual links.

6.2.2.1 Elaboration, and Elaborated Frame Networks

Frames, as previously defined, have an external structure of links to other frames. These are termed elaborations of the frame. The identification of relations (links) to a frame we call elaborating a frame.

A set of linked elaborated frames is termed an elaborated frame network, or EFN. A single EFN corresponds to the knowledge elements and interrelations required to support performance of an enterprise. A course may include instruction to facilitate acquisition of one, or several, enterprises, thus the domain knowledge base for a course may contain one or a set of EFN. A set of EFN is itself an EFN.

There are four kinds of elaborations.

- o attributes, which represent characteristics of a frame;
- o components, which are the constituents of a frame;
- o abstraction, which represents the generality of frames;
- o association, non-hierarchical aggregations of frames.

6.2.2.2 Attributes

An attribute is a labeled set of values from which objects and their properties may take values over time. Attributes define the characteristics of frames. The constraints placed on a particular attribute determine the legal values that may be taken by objects possessing that attribute. For example, an attribute labeled "color" defined for a frame "Apple" may have legal values "red", "green", "yellow". The set {red, green, yellow} associated with the label "color" and the constraint that only one value from the set may be applied to any single object, together define this attribute.

The following operators may be used to define legal values for attributes: the booleans AND, OR, XOR, NOT; the logicals =, <>, <, <=, >, >=; and the range operator.

Attributes take values from the set of legal values defined for the attribute. The value selected during analysis is termed the "initial" value. In addition, storage may also be reserved for a current value.

6.2.2.3 The Component Elaboration

Each kind of frame has its unique component structure. Entities have parts, activities have steps, processes have both events and causes.

Attributes and components are similar in structure. The distinction has to do with independence of existence. A component has independent status, and therefore, can be independently inserted and deleted; an attribute is an essential part of an entity's definition. For example, a "computer" frame might be defined as the aggregate of the attributes "manufacturer", "model", "id #"; and also as the aggregate of the parts "monitor", "circuit boards", "disk drive", "keyboard". A computer without a monitor still retains its identity as a computer (one that lacks a monitor). On the other hand, a computer without a manufacturer violates the definitional character of computer (at least in terms of the schema defined for that knowledge base).

6.2.2.3.1 Entity Components. An entity can be described by its parts. Each part has at least a name, and an associated function. If the entity is a physical object, then each of its parts may also have a location and a graphical description.

A part, of course, can have sub-parts.

6.2.2.3.2 Activity Components. Learning to perform activities is central to maintenance training. Examples of maintenance activities include repair procedures, diagnostic and troubleshooting procedures, and applying safety rules.

An activity consists of one or a series of steps. Steps are performed in a specified sequence, including loops and conditions. Each step has a set of actions associated.

These actions may be performed in a specified sequence (algorithm), including loops and conditions, or they may be triggered by events (heuristics). All actions are represented in the analysis as action + trigger(s) + consequence(s). Triggers and consequences are either sequence data, changes in attributes, time values, or a combination.

Each activity must link to at least one entity which is an actor (performs action and is capable of varying its behavior in the activity based on some internalized knowledge).

A step, or an action, of an activity may itself be an activity, with its own sub-steps.

6.2.2.3.3 Process Components. The understanding of processes plays an important part in maintenance activities, for example in troubleshooting mechanical, electrical, and electronic devices. The constituents of a process are its stages; each stage has an

associated event topology. An event may itself be a process, or in this context, a sub-process.

An event is a transformation of inputs to outputs. Inputs are either attributes, actions, or time values. Outputs are either attributes, actions, time values, or stage transitions. Transformations are either mathematical, logical, or both. An event may be encapsulated within an entity, or be abstract.

Events may be linked together into event topologies. A topology is like a network, except that it need not be fully interconnected. The network structure supports dependencies among events (the output of event A is the input to event B), feedback (the output of event B becomes an input back into event A), and self-regulation. The addition of timing signals as inputs and outputs (these may be relative or absolute) supports synchronization and temporal dependencies. An event topology describes how a process behaves in respect to changes in its inputs.

A stage is a sequentiable phase within a process which has an event topology that is distinct from those of other stages. In other words, a process behaves fundamentally differently in respect to changes in its inputs from one stage to another. A process need have only one stage. Examples of processes with stages are an electronic device, with stages Off, Power-up, and Operating; and reproduction in a seed, with stages Dormant and Active.

6.2.2.4 The Abstraction Elaboration

Abstraction represents the generality of entities, activities, and processes. The levels of abstraction are instance and class, with relationships among classes in a multi-level generalization hierarchy represented by sub-class and super-class links. The abstraction elaboration is exactly equivalent to generalization. An instance represents a specific entity such as a particular object, person, symbol, or place. It may also represent a particular activity, or a particular process. The instance represents the lowest level of abstraction with no subordinates.

A class is an abstract frame that represents general features held in common by two or more frames. Attributes, and their legal values, help define the class. Instances in the class share these attributes.

6.2.2.5 The Association Elaboration

Associations are non-hierarchical aggregations of frames. Unlike the other aggregation elaborations (attributes and components) in which one frame is viewed as the aggregate of the others in the relation; for an association, each frame in the relation may be thought of as an aggregate of the other frames. An example of an

association is a process linked with another process, each of which may be used as an analogy for the other. The relation is the same seen from the point of view of either process.

Classes of Associative Relations. There are numerous relations among things in the world. Only those relations that have instructional value are included in the knowledge representation. These relations either provide meaningful information for prescribing instruction, or combine with other knowledge structures and relations to promote acquisition of an enterprise.

6.2.2.6 Collections

Collections are sets of frames all of the same class. For example, if the class "passenger" were defined, as well as a number of instances of passenger, such as "J. Smith", then a collection of passengers could be defined that would include a group of passenger instances. This collection could be labeled, for example "passengers-flight-75".

Collections, unlike the other relations, are not considered to be elaborations. This distinction, though somewhat arbitrary, is based on the collection being able to be substituted for frames in certain situations. Collections, however, do not have the same status as frames; rather it is the objects of which a collection is formed that are frames. In semantic data model terms, a collection is a grouping relation.

Collections should not be confused with classes, in that they are not more abstract than the frames which form the collection. Hence, there can be no inheritance from a collection to its members. Generalization and inheritance are defined on the class structure of the underlying frame, not the collection. However, collections may have attributes associated with them. These non-inheritable attributes perform functions such as summarizing across the collection. For example, "passengers - flight-75" might have an attribute "count".

The relationship between a collection and its underlying frames is extended to allow defining collections of collections. A sub-collection of "passengers-flight-75" might be "1st-Class-passengers-flight-75".

Collections may take the place of frames in associations. For example, it may be most appropriate to represent that an activity frames "uses" a collection of entity frames, rather than a single entity.

Collections may also take the place of frames as attributes, and as components. For example, the parts of a "computer" entity may include a collection of "memory chips".

A single frame, of course, may be a member of more than one collection. Collection membership is defined by an expression using a syntax equivalent to that for defining class membership.

6.2.3 **Integrating a Simulation Capability**

Knowledge analysis, which is the process of representing subject matter using the knowledge representation model, is relatively straightforward. Knowledge acquisition, which is the process of eliciting the subject matter from an SME in such a form that it can be represented using the knowledge representation model, is more difficult.

The challenge is to find methods of eliciting knowledge from the subject matter experts that are intuitive and couched in terminology and conceptual structures relevant to the domain, but which are sufficiently structured to allow automatic translation to the formalisms of the KR model.

Building a device simulation is also a method for the identification of entity and process components. The use of a simulation editor, such as RAPIDS to produce objects identifies entities and their components; the specification of behavior of objects in effect identifies the events topologies of processes. The simulation editor can be extended so that it generates parts of the EFN knowledge structure automatically as a result of building the device simulation. This would then be augmented by other methods to acquire other elements of the subject matter, such as activity components, and abstraction.

6.2.4 **Example Knowledge Analysis for T-38 Maintenance Training**

This example is for the engine starting procedure.

6.2.4.1 Entities

left engine starting circuitry
right engine starting circuitry
left start button
right start button
left timer
right timer
left igniter
right igniter
left engine
right engine
external power
left generator
right generator
left generator light
right generator light
diverter valve
air
fuel

ignition spark
left AC bus
right AC bus
battery
throttle
front panel
left engine instruments
right engine instruments
left transformer rectifier
right transformer rectifier
static inverter

6.2.4.2 Activities

start right engine
start left engine
on-ground start
emergency start, no engines running
in-air start, one engine running

6.2.4.3 Processes

engine ignition
generation of ac power
diverter valve operation
cross-over system

6.2.4.4 Attributes

diverter valve
state: (centered, left, right)
air-in, air-out-left, air-out-right (numeric)
actuator
force (numeric)

6.2.4.5 Entity Components

left engine starting circuitry
left start button
left timer
left igniter
left generator
left generator light
left AC bus
left transformer rectifier
right engine starting circuitry
right start button
right timer
right igniter
right generator
right generator light
right AC bus
right transformer rectifier

6.2.4.6 Activity Components

```
start engine (right or left)
1 check fore, aft, and under aircraft
2 apply external air
3 depress engine start button 14% rpm
4 advance throttle to idle
5 if fuel flow >= 360 lb/hr and no ignition, retard throttle to
off; wait 2 minutes; go to 3
6 check engine instruments
7 check hydraulic pressure
8 check caution light panel
```

6.2.4.7 Process Components

```
diverter valve operation
inputs: (left_actuator.force, right_actuator.force,
diverter_valve.air_in);
outputs: (diverter_valve.air_out_right,
diverter_valve.air_out_left);
transformation:
(if left_actuator.force >= right_actuator.force then
diverter_valve.air_out_left <- diverter_valve.air_in;
diverter_valve.air_out_right <- 0
else if right_actuator.force >= left_actuator.force then
diverter_valve.air_out_right <- diverter_valve.air_in;
diverter_valve.air_out_left <- 0
else diverter_valve.air_out_left <- 1/2 diverter_valve.air_in;
diverter_valve.air_out_right <- 1/2 diverter_valve.air_in)
```

6.2.4.8 Abstractions

```
engine starting procedures
on ground start
emergency start
in air start
engine
left engine
right engine
engine starting circuitry
left engine starting circuitry
right engine starting circuitry
starting button
left starting button
right starting button
timer
left timer
right timer
igniter
left igniter
right igniter
generator light
left generator light
right generator light
```

AC bus
left AC bus
right AC bus
engine instruments
left engine instruments
right engine instruments
transformer rectifier
left transformer rectifier
right transformer rectifier

6.2.4.9 Associations

process diverter_valve_operation involves entities left_actuator, right_actuator
activity start_engine uses entities air, throttle, start button, fuel, engine_instruments
activity start_engine applies process engine_ignition

6.3 Transaction Authoring

TRX provides a library of reusable instructional programs, or transaction shells, for the delivery of instruction. These programs contain generalized instructional algorithms, each appropriate for teaching a certain type of content, but do not contain any content. Each shell incorporates a number of parameters, configurable by the author, which control the functioning of the shell during course delivery. An instructional transaction (TRX) is a particular instructional interaction with a student. A transaction is characterized as a bounded interchange between an instructional system and a student, which facilitates the acquisition by the student of a specified competence. Transactions comprise the entire range of instructional interactions including: one-way transmission of information (e.g. video, lecture, or document -- which are not very good transactions because they lack interaction); discussions and conversations; tutoring (e.g. traditional CAI and Intelligent Tutoring Systems); simulations; and micro-worlds (with or without coaching).

The effectiveness of a transaction is determined by the extent of the relevant active mental processing required and the nature of the learner's interaction with the content to be learned. An adequate transaction can assume both expository and inquisitory modes; it allows the degree of learner or system control to be adjusted; it includes display and response parameters which allow the transaction to be customized for different learners, different subject matters and different delivery systems. Each transaction is also capable of being invoked to perform different instructional functions with its content: overview; familiarity instruction; basic instruction; example; practice; remediation; and assessment.

A transaction shell (TRXS) is a piece of computer code which when executed causes a given transaction to take place. A transaction

shell knows what knowledge it must have in order to execute its interaction with the learner. It is able to query the domain-knowledge base to find the required knowledge and thus be able to instantiate its knowledge slots. If the domain knowledge base does not contain the necessary knowledge, the transaction shell can direct the user/designer to supply the required content. Once a transaction has been selected or prescribed, it must then be configured and authored. Configuration involves setting the parameters, modifying the strategy, and attaching the content. Authoring involves attaching domain specific instructional materials to the instructional structure set up by the transaction. For example, in concept learning by compare/ contrast, the transaction would contain all the elements to generate examples, practice, and assessment items for concept learning. However, it would require that images of the different concepts, with the defining attributes indicated, be provided by the designer. Each transaction shell knows what domain specific data it requires, and will guide the designer in preparing and entering that data.

Each transaction shell (TRXS) has default values for each of its parameters, including its strategy elements. While most practical instruction will require the modification of these parameters, the acceptance of default values allows the rapid prototyping of instruction incorporating the content and instructional strategies identified. This rapid prototyping allows the designer to get a feel for the structure and look of the finished course while still early enough in the design process to easily change design decisions. The effects of making different design choices can also be easily compared. Transaction shells reside in a transaction library. In addition, configured and authored components are also stored in the library. The library supports the reuse of components, and is a key element in improving the efficiency of the design process.

6.3.1 **Classes of Transaction Shells**

There are several classes of transactions, with each class differentiated from the others in terms of the knowledge structures and performance components instructed. The primary transaction classes are component, abstraction, association, and enterprise. Component transactions instruct all or part of one component hierarchy (parts, steps, or events) in the elaborated frame network. Abstraction transactions instruct all or part of an abstraction hierarchy. Association transactions instruct two or more frames linked by an association relation. Enterprise transactions require as a knowledge base all frames and their interrelations for a given enterprise (Enterprise transactions will be discussed further in the section on Strategy Analysis).

Within each primary class are a number of subclasses. There are 12 subclasses: for components, the subclasses of identify, execute, and interpret; for abstraction, the subclasses of judge,

classify/decide, generalize, and transfer; for association, propagate, analogize, and substitute.

6.3.1.1 Component Transactions

There are three classes of component transactions corresponding to the three types of knowledge frames: identify for entity frames, execute for activity frames, and interpret for process frames.

An identify transaction requires either an instance or class entity frame. It enables the student to acquire the names, functions, properties, and relative location of all the parts which comprise an entity. The student knows what it is.

An execute transaction requires either an instance or class activity frame. It enables the student to acquire the steps of the activity. The student knows how and is able to do the activity.

An interpret transaction requires either an instance or class process frame. It enables the student to acquire the events and causes in a process. This means that the student knows why it works and can explain the events which lead to a given consequence or can predict the consequence from a series of events.

6.3.1.2 Abstraction Transactions

Different types of abstraction transactions can be discriminated on the basis of the performance required and the different combinations of frames from an abstraction hierarchy involved in the transaction. We have identified at least four classes of abstraction transactions: judge, classify/decide, generalize, and transfer.

A judge transaction requires a class frame with two or more subordinate instance frames. These frames can be entity, activity, or process frames. It enables the student to acquire the ability to order the instances of a given class on the basis of some dimension (criterion). The dimensions can be any attribute or combination of attributes. Judging the performance of others as they perform an activity is an example. Ordering a set of objects is an example.

A classify/decide transaction requires a superclass frame with two or more subordinate class frames each of which have two or more instance frames. These frames can be entity, activity, or process frames. It enables the student to acquire the ability to sort or classify instances as to class membership. It enables the student to know when to select one alternative from another. Concept identification is an example. Deciding among alternative activities to accomplish some goal is an example. Editing (selecting the appropriate usage) is an example.

A generalize transaction requires a superclass frame with two or more subordinate class frames each of which have two or more instance frames. These frames can be entity, activity, or process frames. Generalization transactions enable the student to acquire the ability to combine instances of two or more classes into a more general class. Generalization is the inverse of classification.

A transfer transaction requires a superclass frame and one or more class frames. These frames can be entity, activity or process frames. It enables the student to acquire an abstraction model, that is, a generalized set of steps for an activity, or a generalized set of events for a process, and to apply this abstraction model to a previously unencountered class or instance of the activity or process.

6.3.1.3 Association Transactions

Different types of association transactions can be discriminated on the basis of the performance required and the different combinations of frames from a set of associated frames involved in the transaction. We have identified at least five classes of association transactions: propagate, analogize, substitute, design, and discover.

A propagate transaction requires two or more associated frames. The most common relations between knowledge frames, uses, requires, applies -- all involve propagation. A propagation transaction makes a deliberate effort to facilitate the student's integration of information from two or more associated knowledge frames. One of the most important propagation associations is the link between an application activity and a tool activity; another is the link between a method activity and a process. Propagation enables the student to acquire one set of skills in the context of another set of skills. While learning an application activity the student can simultaneously learn the tool activity for doing the application. While learning a tool, the student can simultaneously learn application activities for the tool. While learning a process, the student can simultaneously learn a method activity for studying or observing the process. While learning a method activity, the student can simultaneously learn the process for which the method was devised.

An analogize transaction requires two or more knowledge frames linked by the relation analogy **for**. It enables the student to acquire the steps from one activity by likening it to an analogous activity; or to acquire the events in one process by likening it to an analogous process or activity.

A substitute transaction requires two or more knowledge frames linked by the relation alternative **for**. It enables the student to learn an alternative activity or process by comparison,

elaboration, or extension of a previously learned activity or process. It also enables the student to acquire alternative ways to accomplish a given activity or to explain a given process.

6.3.2 Transaction Shells for T-38 Maintenance Training

An instructional transaction approach to curriculum development enables us to think of the knowledge and skills to be learned at a more integrated level. Rather than thinking of individual skills, it is desirable to identify complex sets of related activities and to build the curriculum around the acquisition of these complex human enterprises. The interactions necessary to promote the acquisition of all of the knowledge and skill associated with a given enterprise, comprises a transaction family.

Henry Halff suggested a "rough list of the tasks that a maintainer must master in order to effectively maintain a piece of equipment." We suggest that this list provides a first cut list of the kinds of human enterprises that an Air Force maintenance curriculum might enable a learner to acquire. Halff's list includes the following enterprise classes: equipment operation, equipment calibration and adjustment, equipment testing, access and disassembly, equipment repair, and troubleshooting. The curriculum consists of a number of specific instances of each of these classes. A more detailed analysis of the curriculum will identify each of these specific enterprises and the knowledge and skill which comprises each of these enterprises.

The type and sequence of interactions necessary to acquire each of these complex enterprises is different. It is proposed that a high level transaction manager be designed and developed for each of the different types of curriculum enterprises. This transaction manager would be a program that could be easily configured to call and sequence the primary transactions identified as necessary for this curriculum. We would propose the development of transaction families to support the enterprise classes of the maintenance curriculum. Halff also suggested that acquiring a model of the equipment was a necessary component of any maintenance training. In addition to transaction families for each of the six maintenance training enterprises identified, we also suggest an equipment model family of transactions which will be a component of each of the other transaction families. In addition he suggested as a possible procedure "redesign and jury rig". This is a complex enterprise for which we have also defined a separate transaction family.

Figure 6.2 identifies the five transaction shell instances which enable learners to construct a mental model of a particular device or piece of equipment. This particular transaction family does not represent a "stand-alone" enterprise, but is a necessary component of the transaction family required for every other maintenance training enterprise. Two classes of

transactions are represented: identify - (1) physical & conceptual structure; and interpret - (2) device functioning, (3) device configuration, (4) fault recognition, and (5) prediction.

Figure 6.3 identifies the two transaction shell instances and the two nested transaction families which enable learners to operate a particular device or piece of equipment. Two classes of transactions are represented: execute - (6) equipment operation procedures; and decide/classify - (7) operation procedure or job aid selection.

Figure 6.4 identifies the three transaction shell instances and the nested transaction family which enable learners to calibrate and adjust a particular device or piece of equipment. Three classes of transactions are represented: execute - (8) calibration and adjustment procedures; judge - (9) calibration and adjustment judgment; and decide/classify - (10) calibrate and adjust procedure or job aid selection.

Figure 6.5 identifies the three transaction shell instances and the nested transaction family which enable learners to test a particular device or piece of equipment. Three classes of transactions are represented: execute - (11) testing procedures; judge - (12) testing judgment; and decide/classify - (13) test procedure or job aid selection.

Figure 6.6 identifies the two transaction shell instances and the nested transaction family which enable learners to access and disassemble a particular device or piece of equipment. Two classes of transactions are represented: execute - (14) access and disassembly procedures; and decide/classify - (15) access and disassembly procedure or job aid selection.

Figure 6.7 identifies the two transaction shell instances and the four nested transaction families which enable learners to repair a particular device or piece of equipment. Two classes of transactions are represented: execute - (16) repair procedures; and decide/classify - (17) repair procedure or job aid selection.

Figure 6.8 identifies the four transaction shell instances and the four nested transaction families which enable learners to troubleshoot a particular device or piece of equipment. Two classes of transactions are represented: execute - (18) logical fault isolation procedures, and (19) intuitive fault isolation procedures; and decide/classify - (20) logical fault isolation procedure or job aid selection and (21) intuitive fault isolation procedure or job aid selection.

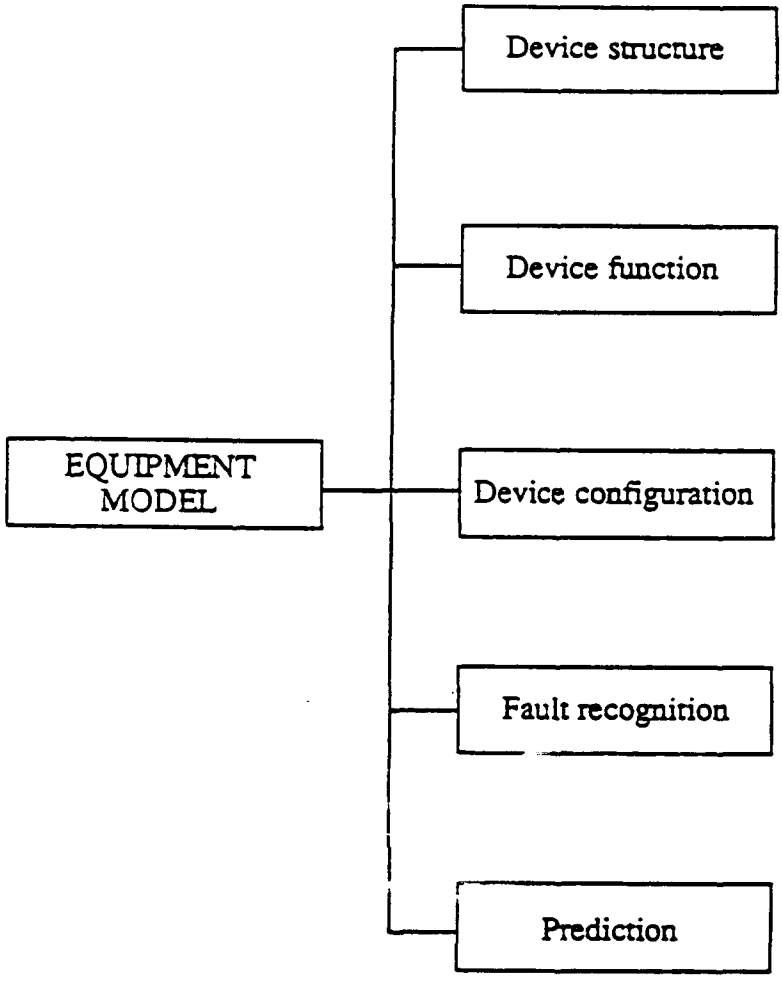


Figure 6.2 Transaction family for acquiring an equipment mental model.

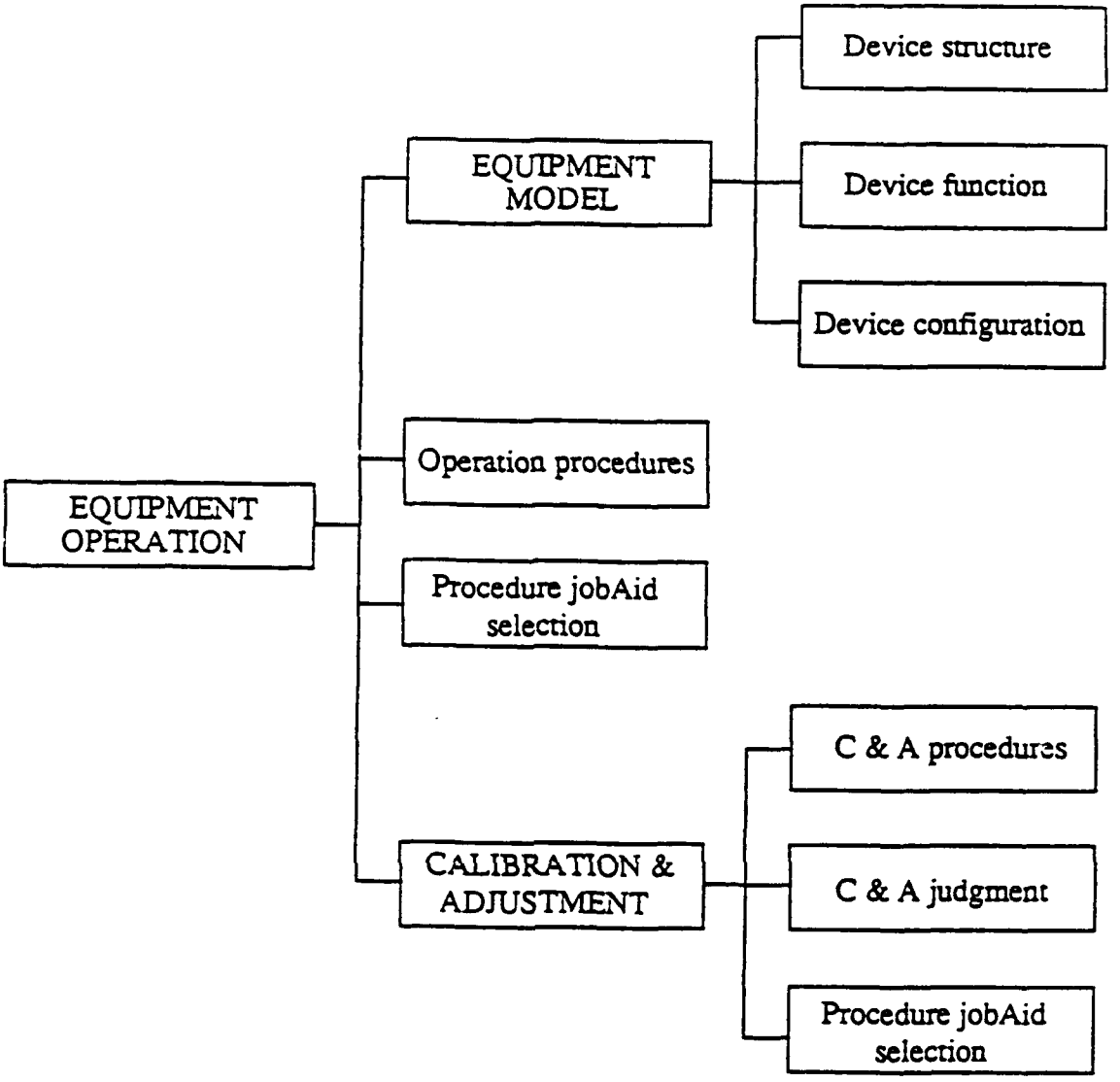


Figure 6.3 Transaction family for acquiring equipment operation enterprises.

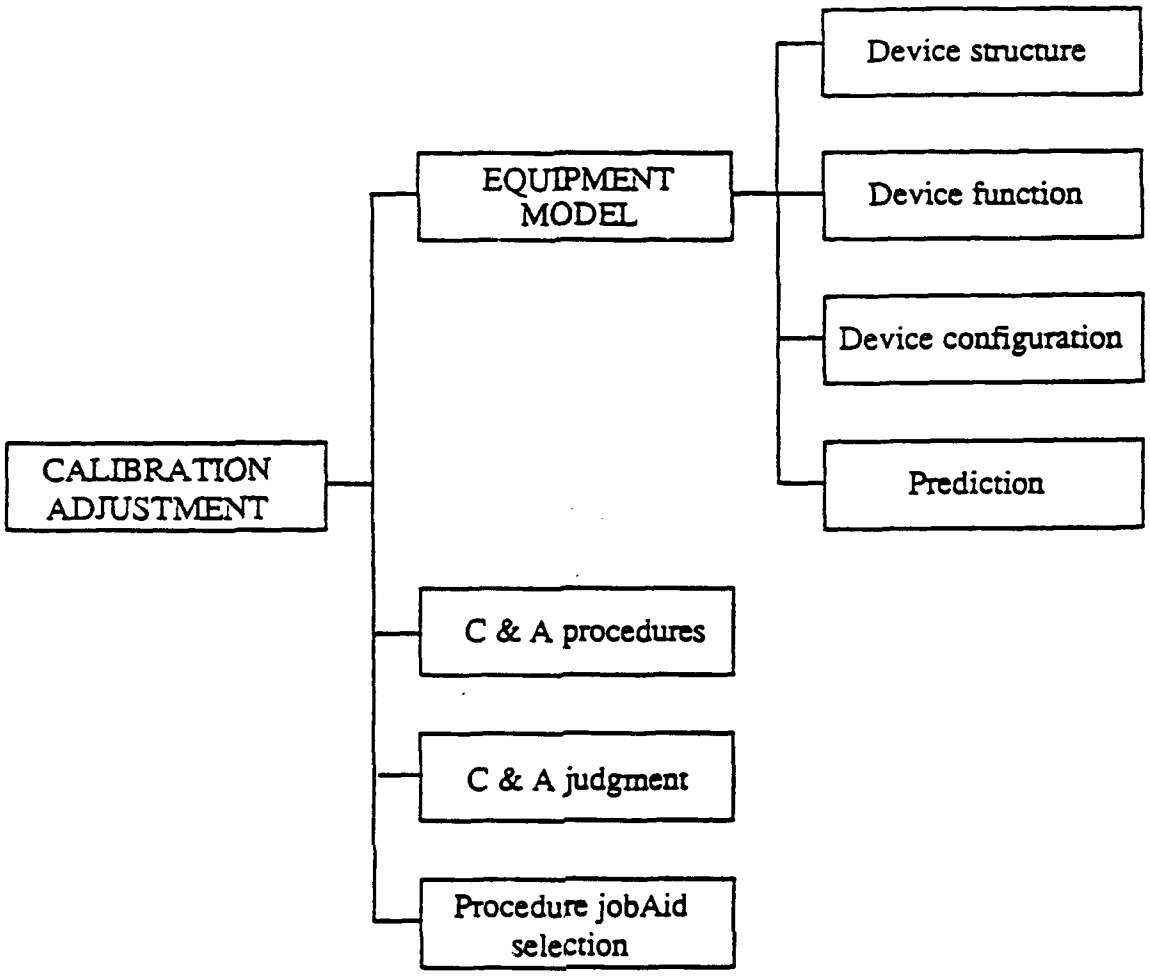


Figure 6.4 Transaction family for acquiring equipment calibration and adjustment enterprises.

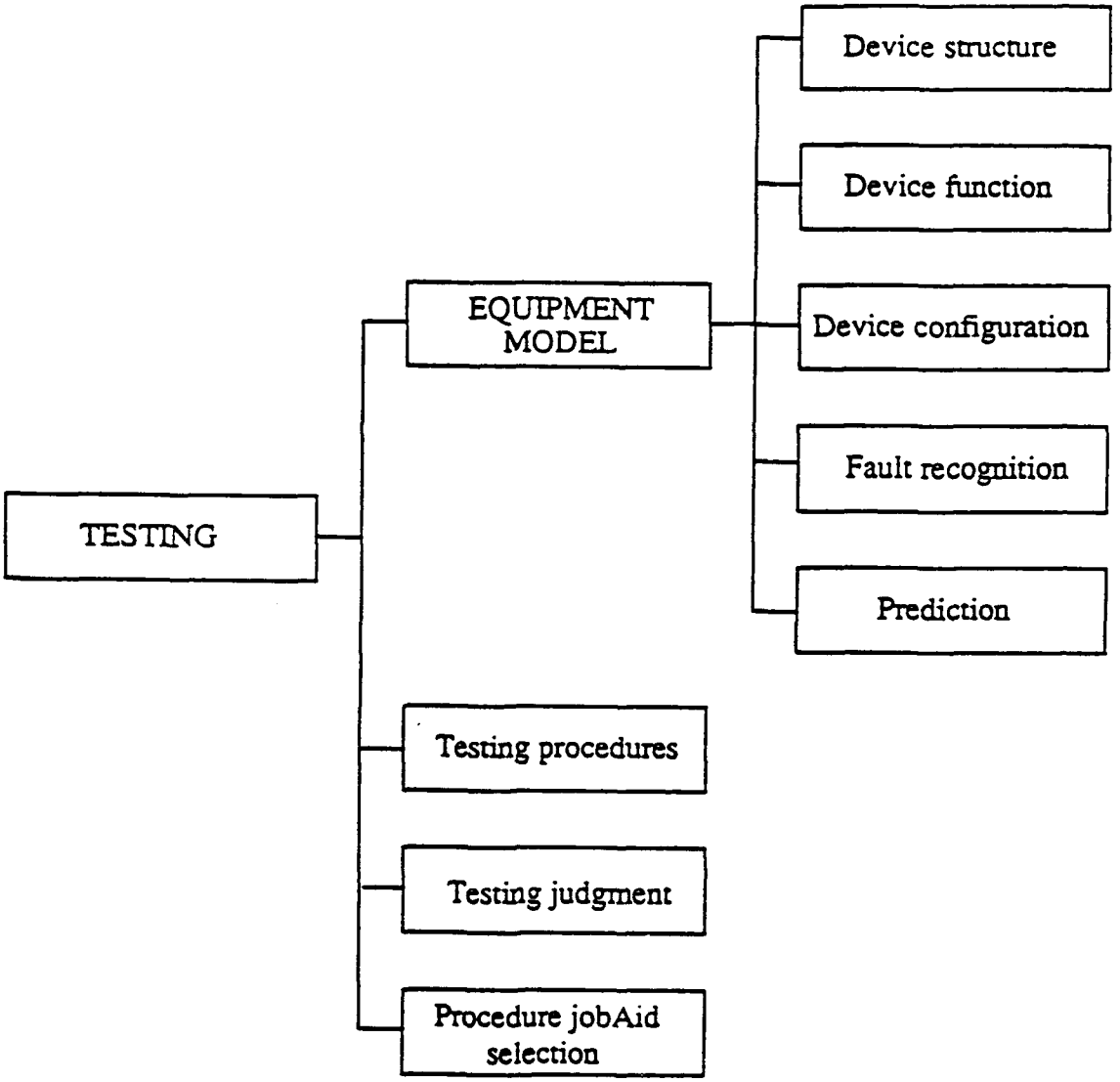


Figure 6.5 Transaction family for acquiring equipment testing enterprises.

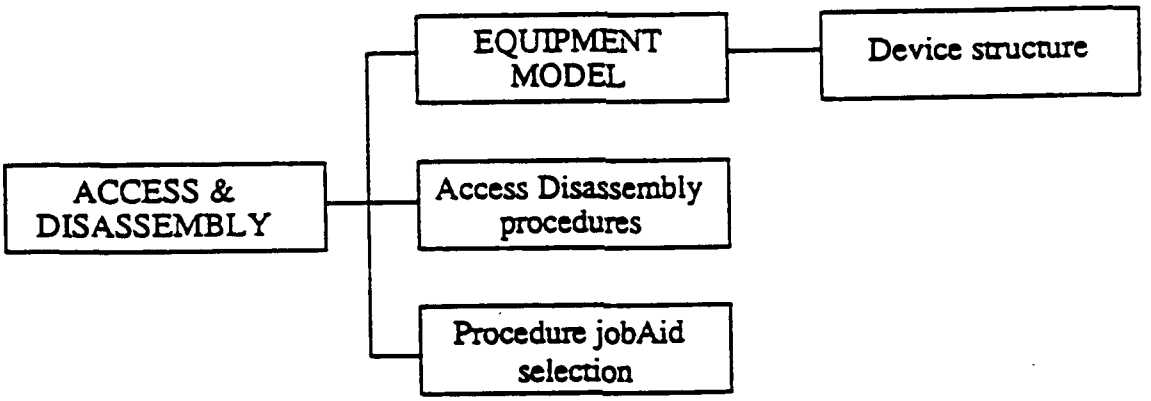


Figure 6.6 Transaction family for acquiring equipment access and disassembly enterprises.

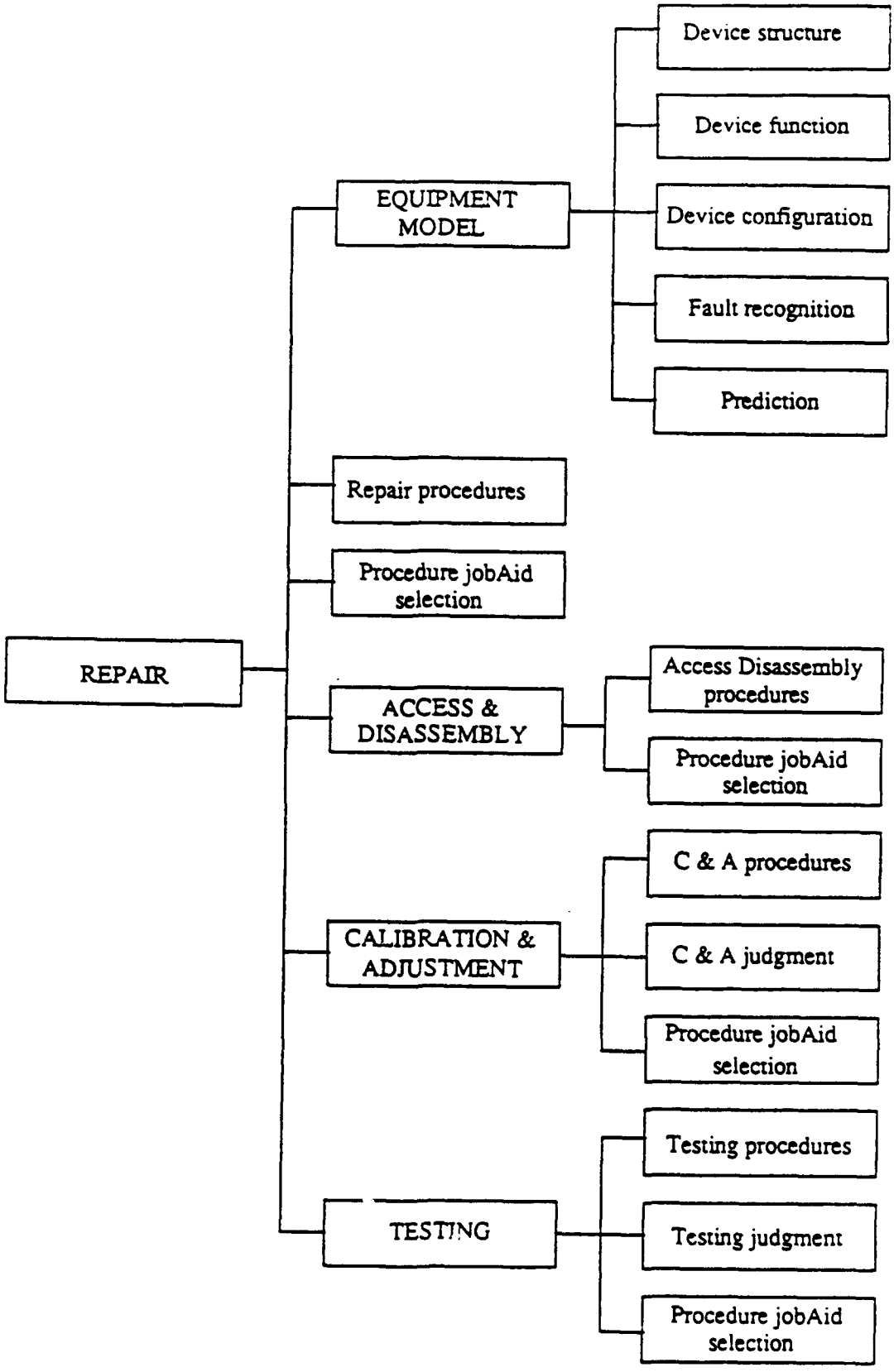


Figure 6.7 Transaction family for acquiring equipment repair enterprises.

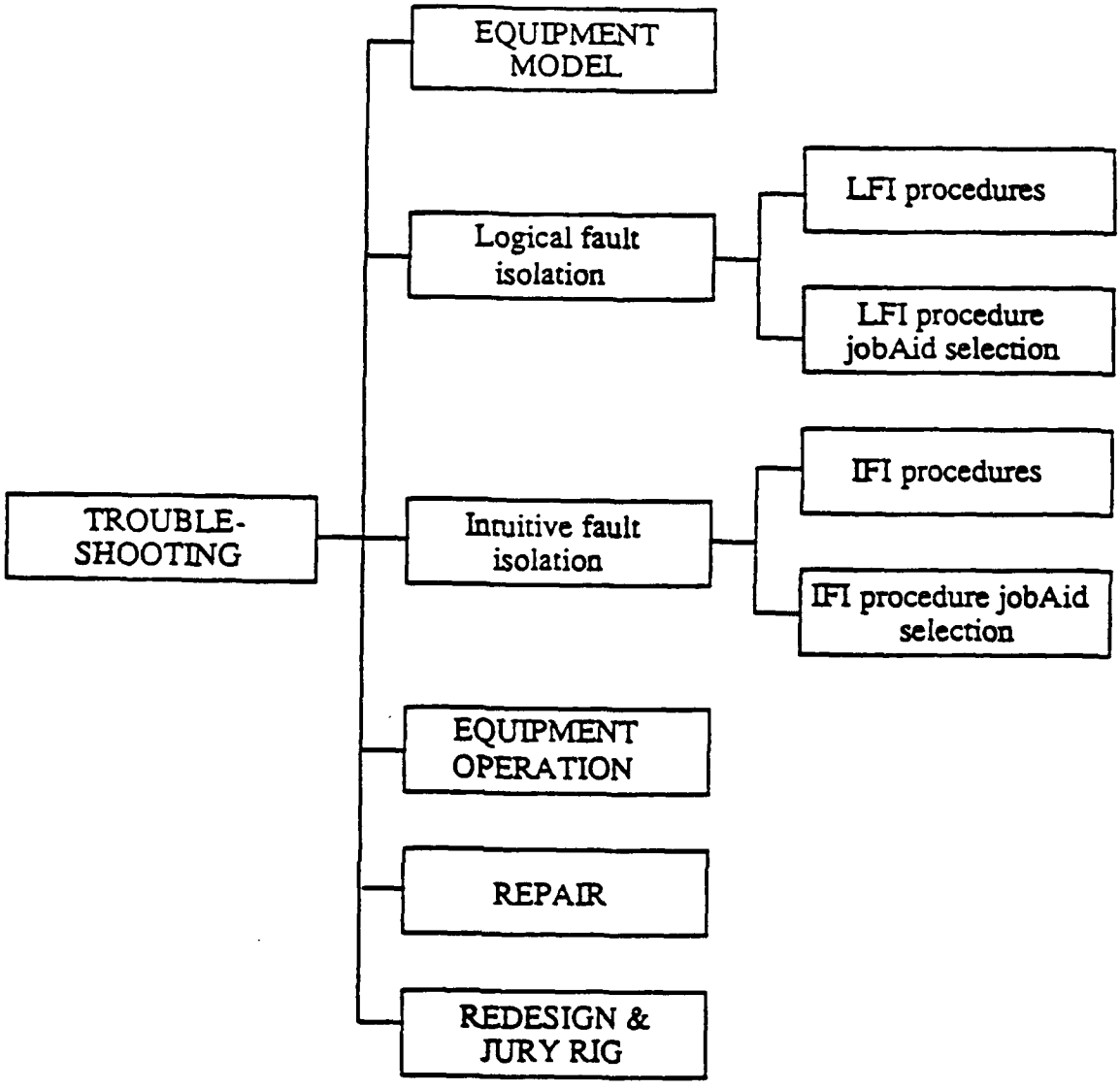


Figure 6.8 Transaction family for acquiring equipment trouble-shooting enterprises.

Figure 6.9 identifies the six transaction shell instances and the three nested transaction families which enable learners to design or jury rig a particular device or piece of equipment. Five classes of transactions are represented: execute - (22) heuristic jury rig procedures; decide/classify - (23) redesign or jury rig procedure selection; transfer - and (24) procedural transfer; analogize - (25) conceptual and procedural analogies; substitute - (26) conceptual, functional, and procedural substitution; and design - (27) redesign techniques.

Merrill previously identified transaction classes. The transactions assigned to each of the transaction families employed in maintenance training have been identified as to transaction class. This classification has implications for the design of maintenance training transaction shells. First, all of the transactions in a given class share common features in terms of the nature of the interaction modes, their interaction strategies, etc. It should be possible to construct a common-base transaction shell for each class and then customize this transaction shell to particularize this shell according to the variations and transaction families of which it is a part. In addition, a particular transaction shell instance may occur in several families. Its interaction modes, interaction strategies, knowledge representation, and interaction parameters will vary depending on the family of which it is a part. Here again some development efficiencies can be realized by designing a common transaction shell and then developing variations of this shell for the various transaction families of which it is a part.

Maintenance training will utilize nine of the twelve transaction classes: identify, interpret, execute, judge, decide/classify, transfer, analogize, substitute, and design. In the following list each of the nine transaction classes is associated with the maintenance transaction families in which it is employed.

Identify:
(1) Physical and conceptual structure
Interpret:
(2) Device functioning
Performance.
Device functioning enables the learner to explain the operation of a given device by knowing the sequence of events, and the conditions under which different events occur.
Knowledge required.
An operational controllable simulation of the device. The simulation should have both functional and some degree of structural fidelity.
Interactions.
The transaction must present the operation of the device via a simulation that enables the learner to change conditions and parameter values and observe the effects on the operation of the device. The transaction must be able to illustrate the operation of the device in a structured manner as well as enabling the learner to manipulate the operation.

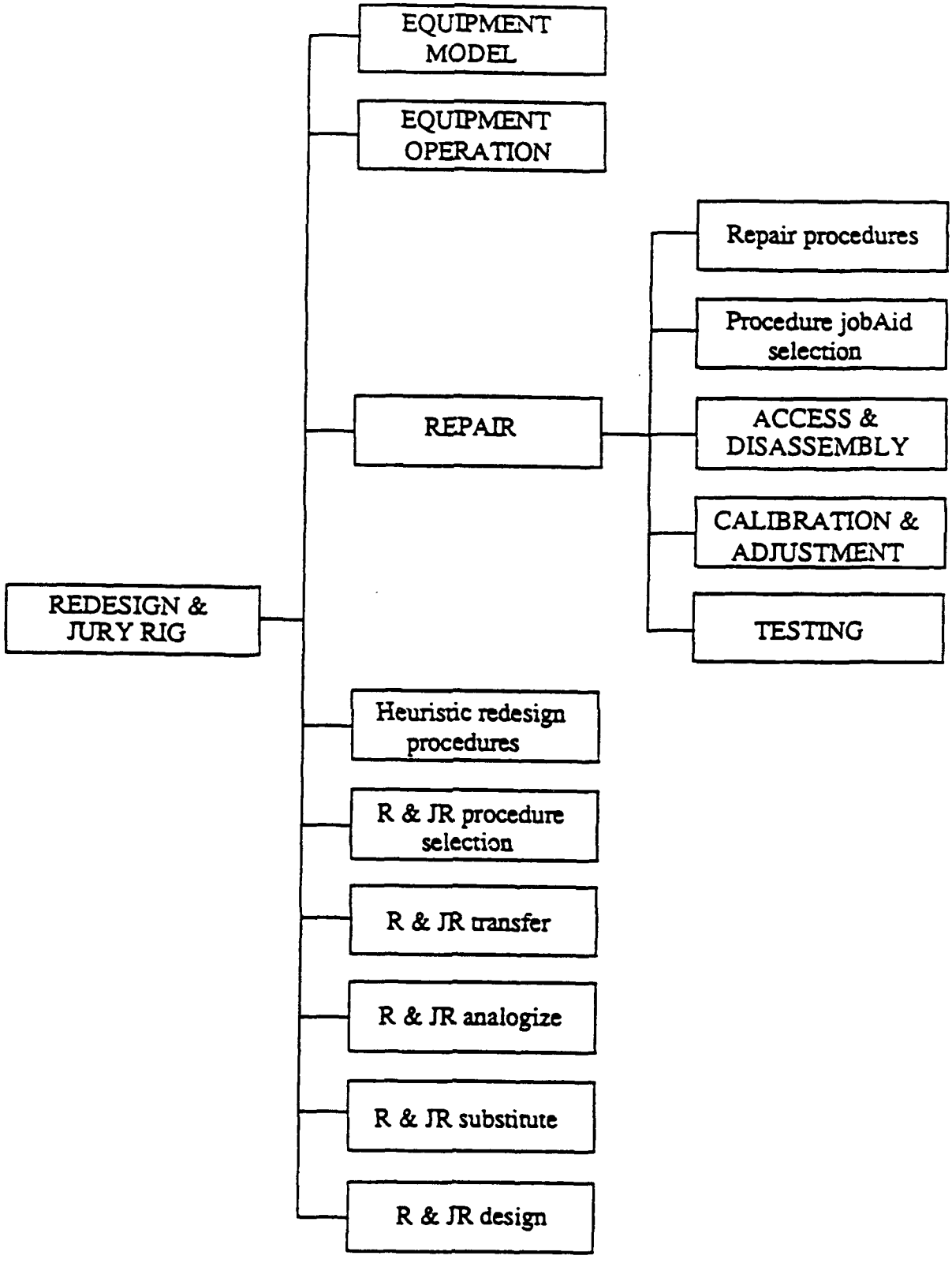


Figure 6.9 Transaction family for acquiring equipment redesign and jury rig enterprises.

(3) Device configuration
(4) Fault recognition
(5) Prediction

Execute:

(6) Equipment operation procedures
(8) Calibration and adjustment procedures
(11) Testing procedures
(14) Access and disassembly procedures
(16) Repair procedures
(18) Logical fault isolation procedures
(19) Intuitive fault isolation procedures
(22) Heuristic jury rig procedures

Judge:

(9) Calibration and adjustment judging
(12) Testing judging

Decide/Classify:

(7) Equipment operation procedure or jobAid selection
(10) Calibration and adjustment procedure or jobAid selection
(13) Test procedure or jobAid selection
(15) Access and disassembly procedure or jobAid selection
(17) Repair procedure or jobAid selection
(20) Logical fault isolation procedure or jobAid selection
(21) Intuitive fault isolation procedure or jobAid selection
(23) Redesign or jury rig procedure selection

Transfer:

(24) Redesign or jury rig procedure transfer

Analogize:

(25) Redesign or jury rig analogies

Substitute:

(26) Redesign or jury rig substitution

Design:

(27) Redesign or jury rig redesign techniques.

The following paragraphs identify the principal performance enabled by each transaction, a preliminary identification of the knowledge base required for each and a preliminary identification of the interaction modes required.

Identify:

(1) Physical and conceptual structure

Learning the names, location, and function of the parts of a device is a prerequisite to learning how a device works or how to operate a device.

Performance.

"Students are shown images of the physical equipment and asked to identify individual components, their function, and their immediate connections" (Halff, p17). Students are shown diagrams representing the conceptual structure of the equipment and asked to identify individual components, their function, and their immediate connections. Students are shown both the physical and conceptual representations and asked to demonstrate the correspondence between these two representations.

Knowledge required.
The knowledge base required includes some representation of the device, probably a graphic representation, with the parts isolated and an associated name and function available.
A graphical representation of the conceptual representation of the system. A pairing of the conceptual symbols with the physical representation of the device.
Interactions.
The transaction must both present the names to the learner and enable the learner to practice locating the parts and identifying the part name and function.
The transaction must present the conceptual names to the learner, must pair the conceptual names with their physical referents.
The transaction must enable the learner to practice identifying conceptual symbols given referents; referents given the conceptual symbols; names of conceptual symbol given its graphic representation; and reproducing the symbol.
Interpret:
(2) Device functioning
Performance.
"Students are asked to discriminate among component states on the basis of some physical depiction of those states" (Halff, p. 17).
The student can explain how the device functions, recognize the various states.
(3) Device configuration
Performance.
"Students are shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state" (Halff, p. 17).
(5) Fault recognition
Performance.
"Students are shown the actual outputs and inputs to an element and asked to determine whether or not the element is faulted" (Halff, p. 17).
(5) Prediction
Performance.
"Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode" (Halff).
Execute:
(6) Equipment operation procedures
Performance.
"Students are required to perform certain operational functions using both a physical and conceptual simulator. That is, each step in the procedure must be executed within the physical simulator and the conceptual simulator. For complex procedures the goal structure of the procedure should be tracked during procedure execution" (Halff).
(8) Calibration and adjustment procedures
Performance.
"Students work with a physical simulation of the device to practice required calibration and adjustment tasks. A conceptual simulation of the system being adjusted or calibrated shows

relations among the components involved in the process" (Halff).

(11) Testing procedures

Performance.

"Students are required to carry out fixed testing procedures on a physical simulation of the equipment. A conceptual simulation of the components being tested is used to exhibit or query the student on the states of these components" (Halff).

(14) Access and disassembly procedures

Performance.

"Students are given the task of gaining access to a particular component. They use a physical simulation of the device to practice the task. A matching conceptual simulation shows which components are accessible at each point in the procedure" (Halff).

(16) Repair procedures

Performance.

(18) Logical fault isolation procedures

Some forms of troubleshooting are best solved by acquiring an accurate operational model of the device or circuit that is malfunctioning and then systematically testing and or replacing components to eliminate potential trouble spots.

Performance.

Logical fault isolation enables the learner to acquire an accurate functional model of the device or circuit. The learner acquires a set of systematic procedures for testing and/or replacing the components of the device or circuit. "Students are provided with a conceptual simulation containing a single faulted component. At each point in the troubleshooting exercise, students would choose an action and exhibit the consequences of the action. The exercise could take many forms. For example, students might be prompted to select actions diagnostic of a particular fault or sets of faults" (Halff).

Knowledge required.

An accurate logical operational model of the device or circuit. A set of malfunctioning components which can be inserted into the device or circuit. A functional simulation of the device or circuit.

Interactions.

The transaction must present the functional model of the device or circuit and enable the learner to use this model in isolating faults that the system inserts into the functional model. The interactions must provide guidance which leads the learner through systematic fault isolation activities.

(19) Intuitive fault isolation procedures

Logical fault isolation is often less efficient than the use of a set of heuristic guidelines (rules of thumb) to identify and correct faults in a device or circuit.

Performance.

Intuitive fault isolation enables the learner to acquire a set of heuristic guidelines and to apply these guidelines in isolating a fault.

Knowledge required.

An accurate logical functional model of the device or circuit. A set of malfunctioning components which can be inserted into the

device or circuit. A functional simulation of the device or circuit. A set of heuristic guidelines for troubleshooting the device or circuit.

Interactions.

The transaction must present the functional model of the device or circuit and enable the learner to use this model in isolating faults that the system inserts into the functional model. The transaction must also present and enable the learner to acquire the heuristic fault isolation rules. The interactions must provide guidance which enables the learner to use the heuristics for fault isolation activities.

(22) Heuristic jury rig procedures

Performance.

"Students are provided with conceptual simulations of tasks requiring complete or partial reconstruction of the equipment. For example, students could be required to restore as much functionality as possible with a limited inventory of spare parts or with other constraints on the reconstruction" (Halff).

Judge:

(9) Calibration and adjustment judging

(12) Testing judging

Decide/Classify:

Performance.

The following performance applies to transactions 7, 10, 13, 15, 17, 20, 21, and 23. Each of these transactions is a minor variation of the same transaction. "Students are asked to identify the procedures needed to deal with particular situations and to select any appropriate job aids. Support is provided for this exercise in the form of subgoals and intermediate steps needed to arrive at the proper selection" (Halff).

(7) Equipment operation procedure or jobAid selection

(10) Calibration and adjustment procedure or jobAid selection

(13) Test procedure or jobAid selection

(15) Access and disassembly procedure or jobAid selection

(17) Repair procedure or jobAid selection

(20) Logical fault isolation procedure or jobAid selection

(21) Intuitive fault isolation procedure or jobAid selection

(23) Redesign or jury rig procedure selection

Transfer:

(24) Redesign or jury rig procedure transfer

Analogize:

(25) Redesign or jury rig analogies

Substitute:

(26) Redesign or jury rig substitution

Design:

(27) Redesign or jury rig redesign techniques.

6.3.3 **Integrating Simulations and Shells**

While many shells will require simulations, these simulations need not be separately created. A single simulation may serve most or all shells for that content. For example, a simulation of the engine starting process can be used by an Interpret shell for teaching device operation, an Execute shell for teaching the

engine starting procedure, and an Identify shell for teaching nomenclature of the entities of the starting mechanism.

6.3.4 **Shell Parameters**

Each shell has a number of parameters which configure the operation of the shell for a particular instructional instantiation. These parameters are set by the instructional designer. Examples of parameters for a naming transaction for the components of an entity include:

6.3.4.1 Focus

Focus is a pointer to an entity frame in the EFN. The component hierarchy in which this frame participates will be instructed by the shell.

6.3.4.2 Content

Indicates how much of the component hierarchy is available for instruction. Takes one of the following (default is All):

All:	entire hierarchy;
(list):	a list of frames (subset of the hierarchy) which are the content.

6.3.4.3 Coverage

Indicates how much of the component hierarchy for the focus is to be instructed. Takes one of the following (default is Focus):

All:	entire hierarchy;
Focus:	the focus, its superpart (the single frame directly above the focus in the hierarchy), and its first level of subparts;
Levels:	requires an additional integer argument, which indicates how many levels of subparts of the focus are instructed;
Exemplar, <label>:	the focus and a single, specified subpart are instructed;
Random Exemplar:	the focus and a single, randomly selected subpart are instructed.

6.3.4.4 Guidance Level

Sets the level of guidance to the learner. Takes one of the following (default is Full):

None:	no guidance;
OnDemand:	guidance presented on learner request only;
Full:	guidance at all times;
Faded:	begin with full guidance, fade to OnDemand by end of transaction.

6.3.4.5 Guidance Type

Takes one of two values (default is Verbose):

Concise: guidance interactions are short and to the point;

Verbose: guidance interactions are detailed and complete.

6.3.4.6 View

Representation of the subject matter. Takes one or more of the following (default is Structural):

Structural: displays the component relation in terms of the knowledge structures, in tree format;

Physical: displays an author-supplied graphic or illustration of the object whose parts are being instructed, representing the physical appearance of the object;

Functional: displays an author-supplied graphic or illustration of the object whose parts are being instructed, representing the functional appearance of the object.

6.3.4.7 Vertical Sequence

Order of introduction of the parts. Takes one of the following values (default is TopDownBreadth):

TopDownBreadth: ordering is from the highest superpart to the lowest level, breadth first (an entire level is introduced before any components of the next level are introduced);

TopDownDepth: ordering is from the highest superpart to the lowest level, depth first;

BottomUp: ordering is from the lowest subpart level to the highest, breadth first.

6.3.4.8 Temporal Sequence

Within the vertical sequence is the order of introduction of the parts on the same level (default is LeftRight):

LeftRight: ordering is according to the representation in the EFN, from left to right;

LowHigh, <attribute label>: ordering is according to the ranking of a named attribute, from low to high;

HighLow, <attribute label>: ordering is according to the ranking of a named attribute, from high to low.

6.3.4.9 Trials

The number of times to sequence through the set of parts. Takes a positive integer value (default is 1).

6.3.4.10 Mastery Level

The percent correct for mastery. Takes a value between 0 and 100 (default is 80%).

6.3.4.11 Response Mode

Type of response performance required of the learner. Takes one of two values, either Recognition or Recall (default is Recognition).

6.3.4.12 Feedback

Timing of feedback. Takes one of the following values (default is PrePractice):

None:	no feedback is given;
PostResponse:	corrective feedback is given immediately after a response;
PrePractice:	corrective feedback is given just before the next opportunity to practice.

6.3.4.13 Replacement

Whenever sampling is used in the transaction, this parameter controls whether a new sample may or may not include items from previous samples (default is With):

With:	a new sample may include items previously used;
Without:	new samples are distinct from previous samples.

6.3.4.14 Items

Whenever a pool of items is practiced or tested, this parameter sets the maximum size of the pool. It takes a positive integer value (default is 3).

6.3.4.15 Timeout

The amount of time to wait for a user response before timing out. If the user response involves typing rather than pointing, the timeout occurs after a base interval, plus a fraction of the base interval multiplied by a number derived from the length of the expected response. If the user response is pointing only, the timeout occurs after the base interval. The value of the parameter is the base interval, a positive integer (default is 3 (seconds)).

6.3.4.16 Item Order

Whenever a pool of items are practiced or tested more than once, this parameter controls whether the ordering is the same or different (default is Random):

Random:	the ordering is random;
Same:	the ordering is fixed.

6.3.4.17 Mode

The mode is the method of interaction with the student. One or more of the following may be selected (default is Overview):

Overview:	presents the knowledge structure from the knowledge base in the Structural view;
Presentation:	presents an author-supplied graphic in either the Physical or the Functional view, and demonstrates the parts to the learner;
Practice:	provides practice for the learner using the author-supplied graphic, in either the Physical or Functional view;
InstanceAssessment:	tests the student's mastery of the material, using the author-supplied graphic, in either the Physical or Functional view.

6.3.4.18 Strategy

Strategy is defined as a sequence of modes. Because modes are fully determined by strategy, the arguments to the Mode parameter are ignored unless the Strategy parameter is None (default is None):

Overview:	the Overview mode;
Familiarity:	the Overview mode followed by the Presentation mode;
Basic:	Overview plus Presentation plus Practice modes;
Mastery:	Overview plus Presentation plus Practice plus InstanceAssessment;
BasicRemediation:	InstanceAssessment, followed by Basic Strategy to remediate errors;
MasteryRemediation:	InstanceAssessment, followed by Mastery Strategy to remediate errors;
Assessment:	InstanceAssessment mode;
Summary:	Overview with Coverage set to Focus, plus Presentation followed by InstanceAssessment, with Coverage set to Random Exemplar for both;
None:	no strategy.

6.3.4.19 Strategic Control

Determines the level of control granted the learner over the selection of strategy, mode, and content. Takes one of the following (default is System):

System:	the strategy, mode, content, and coverage are delivered as set by the parameter values;
Learner:	the learner may select alternate strategies, mode, content, and coverage.

6.3.4.20 Tactical Control

Determines control over the initiation and termination of interactions. Also determines whether the learner may alter the

values of the Guidance, View, and Sequence parameters. Takes one of two values, System or Learner (default is System).

6.3.5 **Configuring Shells**

Configuring is the setting of parameters to a shell, and attaching content.

For example, a call to a shell in the Identify class to instruct the names and locations of the left engine starting circuitry might be configured as follows:

focus:	left engine starting circuitry
content:	all
coverage:	all
guidance level:	faded
guidance type:	concise
view:	structural, functional
vertical sequence:	topDownBreadth
temporal sequence:	leftRight
trials:	1
mastery level:	100%
response mode:	recall
feedback:	postResponse
replacement:	without
items:	3
timeout:	3
item order:	random
mode:	presentation
strategy:	basic
strategic control:	system
tactical control:	learner.

6.3.6 **Detailing Shells**

Detailing is the attachment of graphics and text, prepared off-line, to the knowledge base for use by the shells.

With the use of simulations, detailing requirements are replaced in large measure by simulation authoring.

6.4 **Strategy Analysis**

Strategy Analysis identifies the enterprises to be learned, and selects and sequences transactions to instruct the enterprises. Information about the audience and the instructional setting are gathered in this phase.

6.4.1 **Enterprise Transactions**

An enterprise is a complex human performance that requires an integrated set of knowledge and skills. The goal of instruction is the acquisition by the learner of one or more enterprises. The primary transaction shells, previously described, facilitate the acquisition of the knowledge and skills which comprise

enterprises, but by themselves cannot effect their integration. This integration must be accomplished by transactions at the enterprise level.

An enterprise transaction accomplishes two principal purposes. First, it functions as a transaction manager, providing the overall direction of the execution of the primary transaction instances which instruct the knowledge and skills necessary to the enterprise. Second, it provides for an integration of the learning facilitated by the primary transactions, ideally in the context of a performance or simulation of an authentic activity that is representative of the real-world performance of the enterprise.

The integration is accomplished by focusing the enterprise on a particular performance that is integrative of the elements of the enterprise. The primary transaction that directly instructs the integrative performance becomes the focus transaction for the enterprise. Other transactions are introduced to support the performance on the focus transaction.

A course, then, is defined as a set of enterprise transactions, and their supporting families of primary transactions. A course organization is a nesting and/or ordering of the enterprise transactions.

6.4.1.1 Classes of Enterprise Transactions

Different types of enterprises can be discriminated on the basis of the level of performance required and the type of knowledge involved with this performance. We have identified six classes of enterprises: denote, evaluate, execute, design, interpret, and discover. This class structure may also be used to classify the enterprise transactions according to the class of enterprise being facilitated.

A Denote enterprise transaction requires as a focus one of the following: a primary transaction from the Component class, either an Identify transaction for an entity, an Execute transaction at the Denote level of performance for an activity, or an Interpret transaction at the Denote level of performance for a process frame; or a Classify/Decide primary transaction from the Abstraction class. (Performance level is a parameter to Execute and Interpret transaction shells. For Execute, the values may be either Denote or Perform; for Interpret, values are either Denote, Explain, or Predict.) Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for a denote enterprise is characterized as knowing about something. With a Component class primary transaction as focus, the enterprise transaction enables the student to describe the parts, their functions and locations for an entity; describe the steps for an activity, or describe the events for a process. With a Classify/Decide primary transaction as focus it enables the

student to identify instances or discriminate kinds.

An Evaluate enterprise transaction requires as a focus a Judge primary transaction, of the Abstraction class. The Judge transaction instructs an abstraction hierarchy of either entities, activities, or processes. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for an Evaluate enterprise is characterized as classifying and ranking the adequacy of an entity, the performance of an activity, or the effectiveness of a process.

An Execute enterprise transaction requires as a focus an Execute primary transaction (at the Perform level) from the Component class. The content for the focus transaction is the steps of an activity frame. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for an Execute enterprise is characterized as performing some activity.

A Design enterprise transaction requires as a focus a Design primary transaction from the Association class. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for a Design enterprise is characterized as inventing or creating a new artifact. It enables the student to design a new entity or activity not previously instructed.

An Interpret enterprise transaction requires as a focus an Interpret primary transaction, at the Explain or Predict level of performance, from the Component class. The content for the focus transaction is the events and causal network of a process frame. Primary transactions from the component, abstraction, and association classes would be included to support the focus transaction. Performance for an Interpret enterprise is characterized as knowing why some process works.

A Discover enterprise transaction requires as a focus a Discover primary transaction from the Association class. Primary transactions from the Component, Abstraction, and Association classes would be included to support the focus transaction. Performance for a Discover enterprise is characterized as finding a new relationship or process. It enables the student to discover a new entity or process not previously instructed.

6.4.2 **Authoring Enterprise Transactions**

The enterprise transaction is responsible for integrating the instruction of the knowledge and skills necessary to the enterprise. This integration is accomplished by the selection of a focus transaction that instructs an integrative performance, and by sequencing the focus transaction in conjunction with the supporting primary transactions.

Course organization comprises the sequencing of the enterprise transactions themselves, plus the sequencing of primary transactions within each enterprise transaction.

6.4.2.1 Sequencing Alternatives

There are two dimensions of sequencing at the enterprise level, yielding seven sequencing alternatives. The first dimension, Primary Sequence, includes Encyclopedic, Case Study, and Situational.

The Encyclopedic sequence systematically calls each primary transaction to instruct elements of the content, eventually integrating these at the enterprise level. This type of sequencing is often found in textbooks and reference manuals.

The Case Study sequence presents a sequence of carefully selected examples, scenarios, or cases of the focus transaction and the necessary supporting transactions, with each case being complete in and of itself. The sequence of cases is graded on some dimension, such as familiarity, frequency, or criticality.

The Situational sequence is characterized as on-the-job learning, where instruction is delivered on an as-needed basis. Only that instruction necessary to the immediate task is presented; integration must occur opportunistically. Situational sequence is facilitated by an on-line advisor system and student modeling.

The second dimension, which we call Secondary sequence, includes Elaboration, Prerequisite, and Flat sequence.

Elaboration sequence starts with a simple, representative element or elements of the focus content, and progressively adds layers of detail as the instruction progresses. This is similar in many respects to Riegeluth's Elaboration Theory.

Prerequisite sequence orders elements of subject matter according to their dependency interrelations based on Gagne's learning hierarchies. The focus content is at the top level of the hierarchy.

Flat sequence involves no systematic ordering at the secondary level.

The primary and secondary sequences may be combined into seven approaches to sequencing: elaborated, prerequisite, and flat case study; elaborated, prerequisite, and flat encyclopedic; and situational.

Elaborated case study requires a number of cases of the focus transaction, each complete in and of itself. In our earlier example of a Circuit Functioning enterprise transaction, the focus transaction was an Interpret primary transaction to

instruct circuit functioning. Suppose that the specific enterprise involved the functioning of AC circuits. The enterprise would require a set of cases, each of which would be instructed by the focus transaction. The cases would be drawn from an abstraction hierarchy of circuits, and would be ordered on some relevant dimension, such as complexity, familiarity, frequency of occurrence, etc. Examples might include specific instances of capacitance reactive circuits, resonant circuits, and transformers. Each case would be an instance of a class in the abstraction hierarchy. However, instructing the abstraction hierarchy is not the focus; rather it is the interpretation of circuit functioning which is the focus. The instructing of the abstraction hierarchy is supporting instruction to the focus.

As each case is selected in turn, it is introduced by the focus transaction to the student, following an elaboration secondary sequence. Other information would then be brought into the instruction, from the focus and from supporting transactions, until the circuit had been fully instructed. The next case would then be presented, refreshing and reviewing content that had already been introduced in earlier cases, and introducing additional content.

Prerequisite case study selects cases equivalently, but the secondary sequence follows a prerequisite hierarchy. The case would be overviewed by the focus transaction, but then instruction would build bottom-up following the prerequisites. Each supporting transaction might be called one or more times at different nodes in the hierarchy. Then the next case would be handled in a similar way, refreshing and reviewing content that had already been introduced in earlier cases, and introducing additional content.

Flat case study has no systematic secondary ordering. Once a case has been selected, instruction begins with an overview from the focus; then each supporting transaction would be called in turn to present all required content for that case, finally returning to the focus for a full presentation. Then the next case would be selected.

The encyclopedic sequences are not built on cases. Any abstraction hierarchy is taught as part of the supporting content, rather than being used to generate cases.

Elaborated encyclopedic sequence begins with a representative element or elements of the focus content, introduces supporting content as needed, then builds to the full focus content. Prerequisite encyclopedic sequence begins with an overview of the focus content, then goes to the lowest levels of a prerequisite hierarchy and sequences the primary transactions to deliver instruction for nodes on the hierarchy, building eventually to full focus transaction.

Flat encyclopedic sequence begins with an overview of the focus; then each supporting transaction would be called in turn to present all required supporting content, finally returning to the focus for a full presentation.

Situational sequencing delivers instructional elements on demand, either as a result of user request or based on an online determination by an advisor program of the learning requirements of the user.

6.4.2.2 Making Sequence Decisions

Authoring the sequence for an enterprise transaction involves the following steps:

1 determine enterprise content
2 select the focus transaction
3 for each content grouping, select supporting transactions
4 select primary and secondary sequence
5 if case study, create instances for the focus content
6 identify content for each case
7 if prerequisite sequence, identify prerequisite relations
8 if elaboration sequence, identify elaboration levels
9 configure parameters for each call to a transaction

We now examine the roles of the author and the system for performing each step.

Step 1, determine enterprise content, begins by the author selecting the focus content, such as an activity or process frame. The system then initiates a spreading activation search of the EFN, following relations from that frame. For each relation leading from the frame, the author indicates whether that relation should be included in the enterprise. The search continues from the related frames for any relation that is included; while a path is terminated for any relation not included. This continues until all paths have either reached their end or been terminated. The system then creates the enterprise transaction structure, and stores a representation of the subset of the EFN that has been selected as the content for the enterprise.

Step 2, select the focus transaction, is performed by the author based on a recommendation by the system. The recommendation is based upon the transaction class appropriate for the content structure of the focus.

Step 3, select supporting transactions for each content grouping, is performed by the author with consultation from the system. The system parses the content into content groupings according to the classes of primary transactions (component and abstraction hierarchies, and association links). Each grouping is presented in turn to the author, along with recommended transactions for that grouping. Recommendations are based first on the transaction class appropriate for the content grouping, and may

be further refined by environmental parameters, such as the availability of specific resources. Recommendations may also take into account compatibility with the focus transaction. For example, if the selected focus transaction uses video, employs a particular instructional technique, or is optimized for a given domain area, then the recommended supporting transactions will take this into account. Recommendations are ranked if there is more than one possible choice.

Step 4, select primary and secondary sequence alternatives for the enterprise transaction, is performed by the author based on recommendation of the system.

Step 5, if case study, create instances for the focus content. The author selects an abstraction hierarchy from which cases will be drawn, and identifies the attribute which will be used to order cases. The system then prompts the author to return to knowledge analysis to identify the instances for the classes in the hierarchy which will form the cases.

Step 6, identify content for each case, is performed automatically by the system by parsing the subset of the EFN selected in step 1, selecting any content related to the focus or the instance.

Step 7, if prerequisite sequence, identify prerequisite relations, is performed by the author using a system tool to identify dependency relations. This relation structure is stored with the enterprise. If prerequisite case study, the prerequisite relations for each case may be derived automatically from this structure.

Step 8, if elaboration sequence, identify elaboration levels, is performed by the author. The number of levels, and the content for each level of elaboration, is identified. If case study, this is performed for each case. This data is stored with the enterprise transaction. Secondary content for each level of elaboration will be sequenced by the prerequisite relations, if available, or flat. At this point, all primary and secondary sequencing has been completed.

Step 9, configure parameters for each call to each transaction, is performed by the author. The system brings up the configuration for each call in turn, and presets as many parameters as possible. These include the content for the call, based on the earlier steps, and the values of other parameters based on either student attributes and/or earlier configuration decisions for that enterprise. In addition to the normal configuration capabilities, the author may set a parameter for all calls to the transaction from this enterprise, for all calls to any transaction having the parameter from this enterprise, or for all calls to any transaction having the parameter from this course. (An example of the latter might be setting display or

response parameters to establish a uniform interface across the course.)

6.4.3 **Example Strategy Analysis for T-38 Maintenance Training**

The example is for the engine-starting procedure.

Step 1. Determine enterprise content. The focus content is the abstract activity Start-engine. The remainder of the content would be selected by following links in the knowledge base from that frame.

Step 2. Select the focus transaction. The appropriate transaction is from the primary class Execute.

Step 3. Select supporting transactions. At a minimum, transactions from the classes Identify (for the devices) and Interpret (for the processes) will be required.

Step 4. Select primary and secondary sequence. Case study is selected as the primary sequence, elaboration as the secondary sequence.

Step 5. Create instances for each case. At minimum, three cases are identified: ground start, emergency start, and in-air start.

Step 6. Select content for each case. The author performs this step with guidance from the system, by traversing the EFN. For example, the content for an Identify transaction for the instrument panel would require the names and locations of all instruments used in starting procedures.

Step 7. Identify prerequisite relations. Even though the secondary sequence is elaboration, prerequisite relations are needed for ordering supporting content. The author identifies these relationships among the supporting transactions. For example, knowing the locations and names of the controls would be prerequisite to performing the engine-start activity.

Step 8. Identify elaboration levels. For each case, one or more elaboration levels may be identified. For example, the ground start activity may first be taught with no glitches, then with one or more problems introduced.

Step 9. Configure parameters to each call to a transaction shell. This step is performed by the author using the transaction configuration system.

6.5 **Instructional Delivery**

This section describes the actual delivery of instruction to the student.

6.5.1 **Instructional Modes and Strategies**

The type of transaction and the components of its knowledge base limit the interactions that are possible within a given transaction shell. Different classes of transactions will have different types of interactions. Nevertheless, all transactions should include interactions that are characterized by certain interaction modes. Interaction mode alternatives determine the method of interaction with the student.

Interaction modes assume different values on the form of the interaction (expository or inquisitory) and the degree of learner control involved (learner control or system control). Five interaction modes have been identified: overview, presentation, practice, generality assessment, and instance assessment.

Overview mode presents the knowledge structure, as represented in the EFN. For example, in an Identify transaction, overview would show the parts hierarchy of an entity in tree format. Text instruction may accompany the diagrams. The overview serves as an advance organizer, and as a review.

Example Overview mode for an entity component hierarchy.

Presentation demonstrates and presents the content represented by the knowledge structure, in terms of both generalities and instances. For example, an Interpret transaction for device operation would simulate the operation of the device, explaining the events associated with the process.

Example Presentation mode for an Identify transaction.

Example Presentation Mode for an Interpret transaction.

Practice provides opportunity for the learner to work with the content directly. For example, an Execute transaction for an activity would provide a simulation which could be manipulated by the learner with the consequences of actions displayed. Practice for an Interpret transaction would allow the learner to adjust controls, regulate inputs, and modify the functioning of devices, and to predict the consequences of these actions.

Practice Mode for an Interpret Shell.

Generality and instance assessment test at the generality and instance level, respectively. Test results are recorded by the delivery system, under parametric control of the transaction shell.

Interaction strategy is the combination and sequence of interaction modes available to the student. We have identified seven interaction strategy alternatives: overview, familiarity, basic mastery, basic remediation, mastery remediation, summary, and assessment.

Overview consists of the overview interaction mode.

Familiarity consists of an overview interaction plus a presentation.

Basic instruction consists of an overview plus presentation plus practice.

Mastery instruction consists of overview plus presentation plus practice plus generality and/or instance assessment; if the criterion is not met, a new presentation, practice, and assessment for missed items is engaged until the criterion is met.

Basic remediation consists of generality or instance assessment; if the criterion is not met, then basic instruction is provided for the missed items.

Mastery remediation consists of generality or instance assessment; if the criterion is not met, then mastery instruction is provided for the missed items until the criterion is met. Summary is an overview plus presentation followed by instance assessment; both the presentation and the assessment are for a single representative element of the knowledge structure, rather than the full knowledge structure.

Assessment consists of generality or instance assessment.

6.5.2 **Interaction of Simulations and Shells**

In order to integrate the shells with the simulations at delivery time, there must be a communications interface established. This interface would establish conventions whereby shells would be able to:

- query simulations to determine their capabilities
- query status information from simulations
- issue commands to simulations to set the simulation directly into a state
- replace direct learner input with commands from the shell.

6.5.3 **On-line Delivery Advisor**

The authoring decisions made at design time are based on the designer's best estimate of the learner population. During the delivery of instruction, information about the learner including aptitude, specific goals, motivation, familiarity, and other factors, as well as the learner's expressed preferences, may be taken into account to modify those decisions.

An on-line delivery advisor would have access to the domain knowledge base and the configurations. In addition, it would maintain information about the learner (see Section 9). Using the information gathered about the student, the advisor would adjust design decisions to customize the instruction to more adequately meet the characteristics of the student. The advisor could also engage in a mixed-initiative dialogue with the student which would allow the student to participate in this decision-making.

Example Instruction for T-38 Maintenance Training

Instruction would be extended in transaction shells of the classes Identify, Execute, and Interpret in the following ways:

- Instruction that presented the structure of the knowledge would be incorporated, for example displaying the parts of a device in tree format.

- the parameters governing practice and test would be included, for example item order, replacement, feedback.

Example of pre-practice feedback.

- the capability to invoke a transaction for a specific purpose, such as a single example, or a summary.

Example of a shell being called under Summary Strategy. A single problem is presented to test whether the student needs a refresher.

- system-control presentation, practice, and assessment.

- integration with other transactions via the enterprise transactions, and the structuring of the instruction according to primary and secondary sequencing.

Additional shells would be provided for the classes of Judge, Decide/Classify, Transfer, Analogize, Substitute, and Design. Sample displays from these transactions are shown below.

Example interaction from a Judge transaction.

An interaction from a Classify/Decide transaction for comparing and contrasting related activities.

Example interaction from a Transfer transaction for repair procedures.

SECTION 7. XAIDA FUNCTIONAL BASELINE

7.1 Introduction

The "System/Segment Specification" (DI-CMAN-80008A), one of the data requirements for AIDA Phase II, is described in the DID as the "Functional Baseline" for the system, in this case AIDA. Halff's description of maintenance training for the T-38 engine start system, (see Section 5.2) seems to be the functional baseline or description most specific to the demonstration instructional design (ID) task.

In AIDA Phase I it was agreed that, to gain generality, AIDA would be built around Merrill's second generation theory of instructional design (ID-2) (see Section 6). That means that it must be possible to restate Halff's maintenance training functions in terms of Merrill's ID-2. For example, the concept of the fault tree, widely used in maintenance training and cited by Halff, must be explained in terms of ID-2.

Merrill is clearly the person best able to restate Halff's description of the instructional design task using the language of ID-2. Nevertheless, to get things started, Hickey wrote a function-by-function abstract of Halff's paper. He then undertook to describe each Halff function in Merrill's ID-2 language.

In the following outline, Halff's functions are printed in **bold face** and preceded by an (H); statements from Merrill's ID-2 theory are preceded by an (M). Some Halff functions, like "Describe the system," slip easily into the ID-2 scheme; other Halff functions, like "Develop a hierarchical fault tree," do not.

STAGE 1. KNOWLEDGE ACQUISITION

AIDA will guide the SME to...

7.1.1

H: Describe the system to be maintained.

M: (Develop entity frames as follows.)

M: List the entities of the system to be maintained.

M: List the components (parts) of the entity or system.

M: List the processes of the system to be maintained.

M: List the events and causes of the processes.

T: Using RAPIDS, ...

7.1.2

H: Describe the operator procedure.

M: (Develop activity frames as follows.)

M: List the operator activities, actions or procedures.

M: List the steps in each activity or action.

7.1.3

H: List all possible faults.

M: (Develop process frames as follows.)

M: List all possible faulted processes.

M: List the cause(s) of each faulted process.

7.1.4

H: Develop a hierarchical fault tree. (see Halff's Fig. 2)

M: (How?)

7.1.5

H: **Develop a fault tree interpretation scheme.**

M: (How?)

M: Classify each frame into a class/subclass hierarchy.

M: Link each frame to other frames in the elaborated frame network (EFN).

STAGE 2. DEVELOP INSTRUCTIONAL OBJECTIVES

AIDA will guide the SME/ID to...

7.2.1 H: Develop a mental model.

7.2.1.1 H: Develop a qualitative model.

H: (1) from the bottom up, using a fault tree analysis.

M: (How?)

H: (2) from the top down, by describing the major systems. (a structural breakdown.)

M: (Use results from 7.1.1)

H: (3) combine the results of the bottom-up and top-down analyses.

M: (How?)

7.2.1.2 H: Construct a functional hierarchy (see Fig. 5.5)

M: (Develop process frames as follows.)

M: List all processes.

M: (Elaborate the process frames as follows.)

M: Order the process frames in a systems/components hierarchy (see Fig. 5.5) (see M. 7.2.2.1).

7.2.1.3 H: Construct the images associated with the system and its functions. (This step is not adequately explained.)

M: (How?)

7.2.2 H: Acquire procedural knowledge, using GOMS to develop...

H: (1) a fault tree (see also 7.1.4).

M: (How?)

H: (2) a fault tree interpretation schema (see Table 5.2).

M: (How?)

H: (3) Testing and repair procedures (see Tables 5.1 and 5.3).

M: (Develop activity frames as follows.)

M: List the operator activities or actions.

M: List the steps in each activity or action.

7.2.2.1 **H: Use the fault tree interpretation schema (Fig. 5.3) to retrieve procedures and structural information (from Table 5.1, Fig. 5.3, etc.).**

M: (How?)

7.2.2.2 **H: Represent the elementary procedures...using GOMS.**

M: (Develop activity frames as follows.)

M: List the operator activities or actions.

M: List the steps in each activity or action.

7.2.2.3 **H: Teach the use of the fault tree as follows.**

H: (1) S names the parts (in the declarative mode).

M: (Author the "Identify/Naming" TRX as follows.)

Generate a graphical representation of the fault tree, with each node isolated and an associated name and function available.

H: (2) Then S uses the fault tree to...

(a) retrieve the procedure for checking the functioning of the component (Table 5.2, Line 2.1).

M: "Do you want to see if the component is working?" Y/N____
If Y, the procedure to check out the component is displayed.

M: If the component is not working, then...

(b) retrieve the procedure for repairing a terminal component (Table 5.2, Lines 1.1 and 2.3.1.1).

M: "Do you want to repair the faulted component?" Y/N____
If Y, the repair procedure is displayed.

(c) decide whether or not a component is terminal (Table 5.2, Line 2.3.1).

(d) retrieve successive subcomponents of a non-terminal component (Table 5.2, Line 2.3.2.1).

7.2.3 H: Develop ____????____ for trouble-shooting, in which...

(1) S discovers the appropriate fault isolation strategy.
(2) S learns a problem-solving procedure, comprised of...

(1) context free rules
(2) device-specific rules
(3) procedures for choosing the most information-laden actions.

STAGE 3. DEVELOP INSTRUCTIONAL MATERIALS AND METHODS (TOOLS)

AIDA will guide the SME/ID to...

7.3.1 **H: Develop the infrastructure, to...**

(1) Simulate and describe the equipment to be maintained, in both qualitative and physical terms (using RAPIDS).

(2) Interpret and describe the procedures to be learned.

troubleshooting as problem solving.

7.3.1.1 **Develop the qualitative and physical simulations, i.e., to...**

(1) Develop three types of views:

(a) Global subsystem views.

(b) Views that show all of the components involved in the functions and malfunctions selected for trouble-shooting.

(c) Views that depict only the components involved in particular lower level modes of the fault tree.

(2) Use _____ to generate...

(a) a simulation of all observations that might be made in the course of troubleshooting.

(b) a simulation of all actions.

(c) a simulation of all transitions that refocus the S's attention from one component of the aircraft to another.

Method: SME/ID will walk through every branch of every troubleshooting procedure, noting the views and manipulations needed to support the procedures.

(The physical and qualitative simulations will be linked.)

7.3.1.2 **H: Interpret and describe the procedures to be learned**

AIDA will guide the SME/ID to...

H: (1) Develop a fault tree, in both declarative and procedural terms. (See 2.2.3 above, Author the TRXs.)

H: (2) Develop a fault-tree interpretation schema, using GOMS.

H: (3) the observation and repair procedures attached to nodes in the fault tree, using GOMS.

(With guidance from AIDA, the SME/ID will use an interpreter to computerize these models.)

7.3.1.2.1 Fault-tree interpretation schema

With guidance from AIDA, the SME/ID will develop a display that indicates to the S the following milestones in the fault-tree interpretation schema, or allows the S to indicate these milestones:

(1) S has checked the functionality of the component under consideration (Table 5.2, Line 2.1),

(2) S has isolated and repaired a component (Table 2, Line 2.3.1),

(3) S has completed troubleshooting the subcomponents of a component (Table 5.2, Line 2.3.2.1),

(4) S deals with unsolved cases (Table 5.2, Line 2.3.2.3).

S's progress from one milestone to another will be under three alternative levels of guidance:

(1) Heavily guided practice: computer dictates next milestone to S.

(2) Relaxed guidance: S selects next milestone.

(3) Very relaxed guidance: computer monitors S's actions for consistency with fault-tree interpretation schema.

7.3.1.2.2 Structure of the fault tree

With guidance from AIDA, SME/ID develops both a declarative (graphic display) and procedural interpretation (verbal) of the fault tree.

(1) Develops a graphic display of the fault tree (Figs. 5.2, 5.7).

(2) Develops a computer display which...

(a) describes to the S the procedure s/he should use to check the functionality of any component,

(b) asks the S to designate or execute the procedure,

(c) describes to the S the procedure for repairing any faulted component,

(d) asks the S to designate the procedure.

S can progress under three alternative levels of guidance:

(1) Heavily guided practice: computer dictates next procedure to S.

(2) Relaxed guidance: S selects next procedure.

(3) Very relaxed guidance: computer monitors S's actions for consistency with fault-tree.

7.3.1.2.3 Procedures for observation and repair

With guidance from AIDA, the SME/ID:

(1) Develops a declarative, verbal description of the procedures for repairing faulted components, paraphrasing the GOMS representation.

(2) Develops a representation of the repair procedure in the linked qualitative and physical simulations.

(a) describes to S the procedure S should use to repair any faulted component,

(b) asks S to designate or execute the procedure.

With guidance from AIDA, the SME/ID develops computer-based presentation/practice routines for five attributes of S's performance:

(1) Strategy -- status of a procedure vs. the fault-tree interpretation schema,

(2) Tactics -- status of the procedure vs. the fault tree,

(3) Stepwise descriptions -- the elementary observation and repair procedures,

(4) Conceptual aspects -- the theoretical description in the qualitative model of the procedure. (What does this mean? Does this refer to S's performance?)

(5) Implementation -- the physical actions and observations of the S, in the physical simulation, that implement the observation or repair procedure.

7.3.1.3 Problem-solving and troubleshooting

Guided by AIDA, the ID will consult a mini-expert incorporated in AIDA, either PROFILE or the Fuzzy Rule-Based Model of Hunt and Rouse, and, operating within the RAPIDS-based qualitative simulation,

develop presentation and practice routines for troubleshooting and problem-solving. It will be possible to start the routines in mid problem, particularly when the fault-tree procedure reaches an impasse. The mini-expert will have an explanation facility.

STAGE 4. DEVELOP INSTRUCTIONAL PROCEDURE (CURRICULUM)

AIDA will guide the ID in configuring and assembling the tools listed in Section 3 into a curriculum (multiple lessons) organized in four phases:

(1) System Behavior and Architecture
(2) System Function
(3) Troubleshooting Procedure
(4) Problem Solving

7.4.1 System Behavior and Architecture

Guided by AIDA, the ID will develop exercises with both the qualitative and physical simulations.

Qualitative reasoning tasks will include:

(1) predicting the behavior of individual components,

(2) predicting for the system as a whole the consequences of certain states of the equipment.

Guided by AIDA, the ID will develop the following kinds of exercises: (Are these, or could they be, transaction shells?)

7.4.1.1 Physical and conceptual structure. Students are shown images of the physical equipment and asked to identify (name?) individual components, their function, and their immediate connections.

7.4.1.2 Causal reasoning. Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode.

7.4.1.3 Functional reasoning (a). S is shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state.

7.4.1.4 Functional reasoning (b). S is shown the actual outputs and inputs to an element and asked whether or not the element is faulted.

7.4.1.5 Physical and conceptual appearance. S is asked to discriminate among component states on the basis of some physical depiction of those states.

With guidance from AIDA, the ID will arrange these exercises in a sequence. Some may be embedded in troubleshooting problems.

7.4.2 System Function

Guided by AIDA, the ID will develop exercises based on the functional breakdown of the aircraft (Figure 4.1). Some exercises will provide guided practice in...

(1) system operation (e.g., starting the engines),
(2) elementary observation and repair.

Practice exercises will be arranged in the following order:

(1) qualitative simulation
(2) joint qualitative and physical simulation
(3) physical simulation

Each exercise will begin with a demonstration of the procedure, followed by practice.

Guided by AIDA, the ID will arrange the exercises in a depth-first traversal of the functional breakdown, and to the subgoal structure of the procedures, following the GOMS representation of the procedures.

(2) Guided by AIDA, the ID will create exercises to teach S the tests or observations required to check the functionality of each component of the aircraft.

(a) Qualitative reasoning
(b) Observational procedures
(c) Select and execute test or observational procedures

7.4.3 Troubleshooting Procedures

Guided by AIDA, the ID will develop exercises arranged in the following order...

(1) Depth-first traversal (DT) of the fault tree for each malfunction.

(2) Random traversal (RT) (?) of the fault tree for each malfunction.

AIDA will guide the SME/ID to create exercises in which...

(a) The initial exercises or traversal of each fault tree will provide support for the use of the fault-tree interpretation schema. This support to be withdrawn after practice with a few malfunctions.

(b) Initial exercises with each malfunction will display the fault tree for the malfunction. This support will be faded with practice.

(c) Each exercise will begin in the qualitative simulation. When S masters the fault tree, the physical simulation will be phased in and the qualitative simulation faded.

Documentation available to the S in the field will be available on line.

7.4.4 Problem-Solving Exercises

With guidance from AIDA, the ID will develop problem-solving exercises for S's ready to cope with impasses in the fault-tree method of trouble-shooting. These exercises are to be...

(1) introduced only when S has mastered the fault tree for the malfunction, and

(2) conducted in the presence of a "tutor" working the problem under the same initial condition as the S.

7.4.4.1 Troubleshooting exercises

S is presented with a qualitative simulation containing a single faulted component. At each point in the troubleshooting exercise, the S is prompted to choose an action and exhibit the consequences of the action.

7.4.4.2 Reverse troubleshooting exercises

S is told that a particular component is faulted and is asked to predict the results of designated observations.

7.4.4.3 Case studies

S is given real case studies of intractable trouble-shooting problems.

7.4.5 Typical Troubleshooting Curriculum

Lesson 1. A set of reverse troubleshooting (RT) and troubleshooting (TS) problems that cover the major topological patterns in the device, each pattern addressed first by RT exercises, followed by TS exercises.

Lesson 2. A set of RT and TS problems that cover the equipment's mission-critical faults and their nearest neighbors. S first RTs each major fault and its neighbor, then troubleshoots the pair.

Lesson 3. A repetition of Lesson 1 without RT.

Lesson 4. A repetition of Lesson 2 without RT.

Lesson 5. A mixture of Lessons 3 and 4.

SECTION 8. XAIDA SYSTEM DESIGN

8.1 Introduction

The design of XAIDA is described in detail in the Data Base Design Document (DBDD) and summarized in this section.

The data base identified as the Knowledge Base (KB) for the Advanced Instructional Design Advisor (AIDA) expert system (AIDA.KB), is the repository of the knowledge required to support the AIDA expert system (ES). It consists of the following sub-KBs:

a. Domain Knowledge Base (DOMAIN.KB)
b. Transaction Knowledge Base (TRANSACTION.KB)
c. Enterprise Knowledge Base (ENTERPRISE.KB)
d. Student Knowledge Base (STUDENT.KB)
e. Environment Knowledge Base (ENV.KB)
f. Task Knowledge Base (TASK.KB)
g. Session Knowledge Base (SESSION.KB)

8.2 The Knowledge Base Management System (KBMS)

A knowledge base management system (KBMS) contains the composite software for storing, accessing, manipulating, reasoning, and otherwise controlling the knowledge embodied in the expert system (ES). In AIDA, the KBMS will manage (1) the knowledge that defines and drives the instruction configuration and authoring functions, (2) the instruction configuration and authoring knowledge acquired from subject matter experts (SME) and instructional designers (ID), and (3) the knowledge that defines and drives the instruction delivery functions.

XAIDA will be implemented within an expert system shell (ESS) approved by ALHRD, rather than being developed from scratch using a conventional programming language. The knowledge base manager (KBM) is the core component of the KBMS which controls the base level storage of the knowledge base. AIDA will employ the knowledge base manager (KBM) provided by the approved ESS.

The knowledge base definition language (KBDL) is the "front-end" syntax used for a specific KBMS to define/declare the objects in the knowledge base. The KBDL proposed for the AIDA KBMS will take a generalized, object-oriented (OOD) approach (using frames as objects). The specific KBDL will be the one provided by the approved ESS.

The knowledge base query language (KBQL) is the "front-end" syntax used for a specific KBMS to insert, retrieve, update, and delete the objects in a knowledge base. AIDA will employ the specific knowledge base query language (KBQL) provided by the approved ESS.

The following general terms are herewith defined for use in the description of the AIDA knowledge base structure:

a. knowledge base (KB): The structured knowledge stored in an expert system. In AIDA, a named collection of related frames viewed as a unit.

b. object: A data structure that contains all the information related to a particular entity. It might be considered a frame with additional features allowing it to contain and invoke methods and to send and receive messages.

c. frame: A named set of one or more related slots; equivalent to a record in conventional data base terminology. Frame is the object in this OOD.

d. slot: A named set of one or more related facets; sometimes equivalent to a data field in conventional data base terminology.

e. facet: A variable or a parameter name designating an attribute (value, constraint, link, procedure, etc.) of a slot; sometimes equivalent to a data item in conventional data base terminology.

f. value: The lowest level item of knowledge stored in a KB.

8.3 **Knowledge Base Structure**

AIDA is comprised of 6 major subsystems and 7 knowledge bases as illustrated in Figure 8.1. The Knowledge Acquisition/ Representation, Strategy Analysis, Transaction Authoring and Instruction Delivery systems closely correspond to the steps in the course development process. The Evaluation System further supports the cyclic nature of this process by providing the capability to analyze an instance of a configured and authored course or lesson. The Courseware Generation System may provide the capability to generate courseware for instruction delivery systems external to AIDA at some future date (e.g., ATS, ISS, MERLIN, etc.).

a. The Knowledge Acquisition/Representation System (KARS) interacts with the SME and/or the ID to gather information on (1) the task to be learned, (2) the student who must learn this task, (3) the environment in which the student will be instructed, and (4) a model of the subject matter associated with the task to be learned, as well as supporting instructional material such as figures, diagrams, and descriptions. This knowledge is stored in the Task, Student, Environment, and Domain Knowledge Bases. The KARS may provide the capacity to accept inputs from an external cognitive task analysis system at some future date, such as the ALHRD/MO cognitive task analysis system.

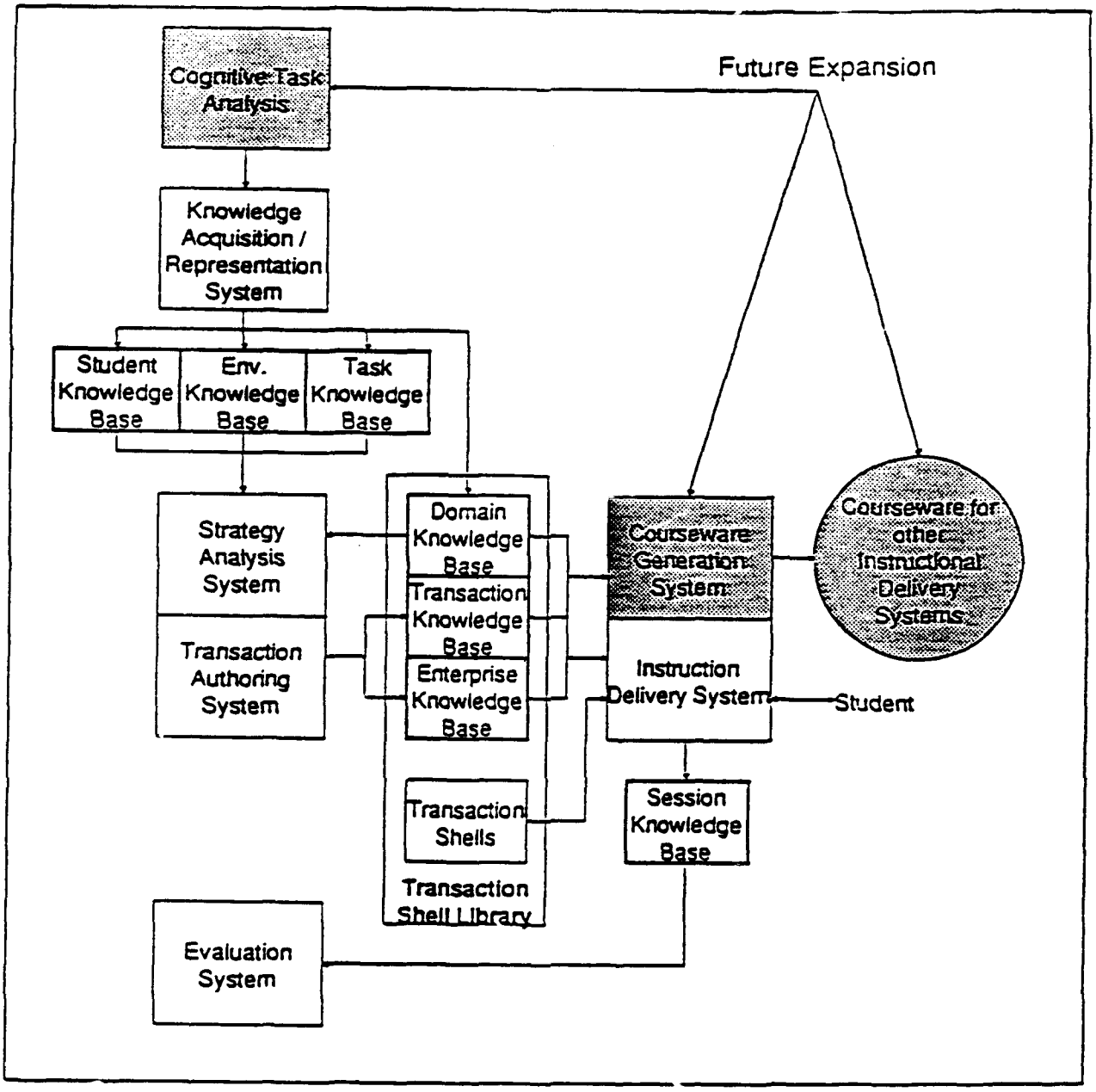


Figure 8.1 AIDA System/Knowledge Base overview.

b. The Strategy Analysis System analyzes the content of Student, Environment, Task, and Domain Knowledge Bases then invokes the Transaction Authoring System to interact with the SME/ID to specify the approach, organization, and additional content of the curriculum, course, or lesson. The approach and organization is stored in the Transaction and Enterprise Knowledge Bases. The content is stored in the Domain Knowledge Base.

c. The Transaction Authoring System contains a library of reusable instructional programs, or transaction shells, for the delivery of instruction. These programs contain generalized instructional algorithms, each appropriate for teaching a certain type of content, but do not contain any content. A transaction shell is a piece of computer code which when executed, causes a given transaction to take place. Each shell incorporates a number of parameters, configurable by the author, which control the functioning of the shell during course delivery. Each shell knows what knowledge it must have in order to execute its interaction with the learner. It is able to query the domain knowledge base to find the required knowledge, and thus be able to instantiate its knowledge slots. If the domain knowledge base does not contain the necessary knowledge, the transaction shell can direct the SME/ID to supply the required content. Once a transaction has been selected or prescribed, it must then be configured and authored. Configuration involves setting the parameters, modifying the strategy, and attaching the content. Authoring involves attaching domain specific instructional materials to the instructional structure set up by the transaction. Each transaction shell has default values for each of its parameters, including its strategy elements.

d. The Instruction Delivery System. Authoring decisions made at design time are based on the designer's best estimate of the student population. During the delivery of instruction, information about the students, their aptitudes, specific goals, motivation, familiarity, and other factors, as well as their expressed preferences, may be taken into account to modify those decisions. The on-line delivery advisor has access to the domain knowledge base and the configurations. In addition, it maintains a student model that contains information about the student. Using the information gathered about the students, the advisor adjusts design decisions to customize the instruction to more adequately meet the characteristics of the student. The advisor also can engage in a mixed-initiative dialogue with the students which allows them to participate in this decision-making.

The basic unit of instruction delivery is the interaction with the students. Transactions are comprised of interactions that are characterized by certain interaction modes, methods of interaction with the students. Interaction

modes assume different values on the form of the interaction (expository or inquisitory) and the degree of learner control involved (learner control or system control). Five interaction modes have been identified: overview, presentation, practice, generality assessment, and instance assessment. Interaction strategy is the combination and sequence of interaction modes available to the students. There are seven interaction strategies: overview, familiarity, basic, mastery, basic remediation, mastery remediation, summary, and assessment.

e. The Evaluation System. AIDA needs both author management and course management systems. Without computer-managed instruction there can be no benchmarks with which to determine progress in the evolution of the AIDA system.

Furthermore, if XAIDA is to be a research tool, then measurement and evaluation issues must be confronted directly by adding a data gathering capability. The built-in evaluation system will enable users to turn in data to be used in refining AIDA. Data will be collected on both instructors and students.

The data gathered will be useful for both formative and summative evaluation of the performance of the SME/ID in authoring courseware, particularly the benchmark lesson on restarting the T-38 jet engine. It will make it possible to generate statistics such as the ratio of author hours to student contact hours.

f. Courseware Generation System. Delivery of instruction to a large number of students may not be a function of the AIDA system. Instruction delivery is more likely to be accomplished via a multi-terminal, multi-media system optimized for instruction delivery, such as ISS, WISE, QUEST, PILOT, TENCORE, etc. The Courseware Generation System analyses the configured and authored transactions and generates courseware in a format that can be used by other instruction delivery systems. XAIDA will, however, have a student mode which will enable the SME/ID to view the instruction sequence as seen by the student in order to evaluate it.

8.3.1 **Domain Knowledge Base (DOMAIN.KB)**

The function of the Domain Knowledge Base (DOMAIN.KB) is to represent "all the knowledge required for instruction leading to [the] acquisition of an integrated human performance, or enterprise." This knowledge includes the cognitive model(s) of the domain knowledge as well as the instructional material associated with that knowledge.

The three kinds of domain knowledge are entities, activities, and processes. For each of these three kinds of domain knowledge, the DOMAIN.KB must specify the following:

a. Attributes: Attributes represent the characteristics of an entity, activity, or process such as its name and function. In the DOMAIN.KB, attributes also represent instructional material such as figures and explanations.

b. Components: Components "represent the constituents of a frame. For an entity, the components would be parts of the entity; for an activity, steps; and for a process, events and causes."

c. Abstractions: Abstractions represent the class/sub-class hierarchy into which an entity, activity, or process may be classified.

d. Associations: Associations represent other associations (relationships) that an entity, activity, or process can have with other entities, activities, and processes. These include relations such as (1) analogy for, (2) alternate for, (3) uses, (4) involves, and (5) applies.

The DOMAIN.KB is self-contained and does not reference other knowledge bases. It is comprised of the following frames:

(1) The Domain Knowledge Frame is the basic unit of the Domain Knowledge Base. It describes content, which can be an entity, activity or process, and specifies the following:

The Attributes Frame, which specifies the attributes of the entity, activity, or process;

The Components Frame, which specifies the component hierarchy to which the entity, activity, or process belongs;

A list of Abstraction Frames, which specify the class/sub-class hierarchy to which the entity, activity, or process belongs; and

A list of Association Frames, which specify the associations (relations) of the entity, activity or process to other entities, activities and processes.

(2) Entity Attributes Frame - specifies the attributes of entities: e.g., Name, Description, Location, and Figure.

(3) Activity Attributes Frame - specifies the attributes of Activities: e.g., Name, Description, Event, and Sequence.

(4) Process Attributes Frame - specifies the attributes of Processes: e.g., Name, Description, Inputs, Outputs, Transformations, Events, and Timing.

(5) Components Frame - specifies the component hierarchies of entities, activities, and processes.

(6) Abstraction Frame - specifies the class/sub-class hierarchies of entities, activities, and processes. The Abstraction Frame can also be used to represent "Collections".

(7) Association Frame - Together, the frames in the DOMAIN.KB are organized into an "Elaborated Frame Network".

8.3.2 **Transaction Knowledge Base (TRANSACTION.KB)**

Transaction Knowledge Base (TRANSACTION.KB) contains descriptions of instructional transactions, where a transaction (TRX) is defined as a "particular interaction with a student." For each transaction, the TRANSACTION.KB will specify the following:

> Transaction Shell (TRXS): When executed, the TRXS causes a particular transaction to take place.
>
> Focus: The focus is the domain knowledge required to perform a particular interaction.
>
> TRXS Parameters: TRXS parameters configure the operation of a TRXS. The parameters are Content, Coverage, Guidance Level, Guidance Type, View, Vertical Sequence, Temporal Sequence, Trials, Mastery Level, Response Mode, Feedback, Replacement, Items, Timeout, Item Order, Modes, Strategy, Strategic Control, and Tactical Control.

The Transaction Knowledge Base is comprised of Transaction Frames and Transaction Shell (TRXS) Parameter Frames. These describe a transaction by specifying (1) the Transaction Shell (TRXS) which will perform the transaction, (2) the focus knowledge of the transaction, and (3) the parameters specifying the performance characteristics of the particular transaction.

8.3.3 **Enterprise Knowledge Base (ENTERPRISE.KB)**

The Enterprise Knowledge Base (ENTERPRISE.KB) represents Enterprise Transactions. An enterprise is a complex human performance that requires an integrated set of knowledge and skills. The goal of instruction is the acquisition by the learner of one or more enterprises. Integration must be accomplished by enterprise transactions. The transactions necessary to acquire all the knowledge and skill associated with a given enterprise comprise a transaction family. In maintenance, the six enterprises are equipment operation, equipment calibration and adjustment, equipment testing, access and disassembly, equipment repair, and troubleshooting.

A high level transaction manager (TRXM) is required for each of the enterprises. The transaction manager is a program that calls

and sequences the primary transactions identified as necessary for a particular curriculum.

In addition to transaction families for each of the 6 maintenance training enterprises, an equipment model family of transactions is a component of each of the other transaction families.

A course is a set of enterprise transactions and their supporting families of primary transactions. A course organization is a nesting and/or ordering of the enterprise transactions.

8.3.4 **Student Knowledge Base (STUDENT.KB)**

XAIDA will be designed with basic information about the students already in the system. The instructional designer will be given the performance capabilities to be acquired by the student, e.g. name the parts. XAIDA will then select and configure transaction shells appropriate to the specified capabilities. The instructional designer will then be prompted to enter any needed content knowledge, e.g., from the T-38 starting system, to complete a frame appropriate to the particular knowledge type.

8.3.5 **Environment Knowledge Base (ENV.KB)**

Because we are assuming a fixed environment (small class, computer-based instruction, located at a TTC), information about the instructional environment will be hard-coded into the XAIDA EXECUTIVE for the time being.

8.3.6 **Task Knowledge Base (TASK.KB)**

The first step in the design of any instruction is a task analysis to determine what should be taught. In XAIDA we are focusing on teaching procedures for maintenance training. Therefore, we will customize an enterprise analysis pertinent to that domain and also customize an elaborated frame network shell pertinent to maintenance training procedures. At some future time, however, the KARS in XAIDA will be enhanced to include a Task Analysis capability. To insure future compatibility, the KARS in XAIDA will be developed around the concepts of cognitive task analysis, as described below.

A GOMS (Goals, Operations, Methods, and Selection Rules) analysis, in which the tasks to be accomplished are broken down into a meaningful series of goals and subgoals. Goals represent a person's intention to perform a task, subtask, or single cognitive or physical operation. Operations characterize elementary physical actions (e.g., pressing a button, setting a switch, or attaching a probe) and cognitive or mental operations (e.g., perceptual operations, retrieving an item from memory, or reading a voltage and storing it in working memory). Methods generate sequences of operations that accomplish specific goals or subgoals. Selection rules determine which method to select.

One way to determine the content of instructional materials and training procedures is to do a complete cognitive simulation of a given task. The advantage of a simulation is that it insures that the analysis is complete.

The most accurate way of determining the mental model to be taught would be to do a complete cognitive simulation. This is not always feasible. Kieras has described three heuristics that can be used to determine the mental model that should be taught in lieu of a complete simulation.

- relevance to task goals
- accessibility to use
- critical procedures and inference strategies

To carry out the first heuristic, an explanation hierarchy is constructed. The second heuristic, "accessibility to use", implies that the device illustration or simulation which is presented to the technician should not contain parts which cannot be accessed. Again this involves mapping the GOMS analysis, but onto the device description, rather than the explanation hierarchy. The third heuristic says that the GOMS analysis should be examined for procedures that will be difficult to learn due to what appears to be arbitrary content.

To decrease the workload of authoring the simulation and/or doing the GOMS analysis for a given domain, XAIDA will contain a library of generic low-level procedures, such as testing igniter plugs. These modules can also be given as screening tests to insure that these low-level methods are learned before entering simulations at a higher level or aimed at specific problems.

Kieras has defined a language call (NGOMSL) or "Natural" GOMS Language. The results of an NGOMSL analysis are implemented in a working simulation. The device knowledge necessary to carry out the simulation will be represented using Anderson's PUPS system, which is compatible with the Transaction Shell representation. Anderson's PUPS (Penultimate Production Systems) theory holds that procedures are acquired by compiling declarative knowledge. The declarative knowledge necessary for compiling the procedures which model the task performance is represented in schema-based structures called PUPS structures. These schema include slots for the function of the entity being represented by the schema, a form slot for the physical appearance of the entity, and a precondition slot which states the preconditions necessary for the function to be achieved. In compiling the productions which are the basis of procedural knowledge, the function slot maps to the goal to be achieved which will require knowledge of the entity represented; the preconditions slot maps onto the condition of the condition-action pair in a production. The form slot in the PUPS tutors holds the form of the current action to be carried out such as a particular LISP function. A similar scheme could be used for representing the GOMS analysis. Merrill has proposed an activity frame that has paths or sequences of

actions. This frame could also have slots for the function, the operators, and the outcome. The values for these slots could probably be automatically generated from a NGOMSL analysis just as it is technically feasible to generate a running production rule-based simulation from an NGOMSL analysis.

The representation scheme proposed by Merrill for AIDA will be used to represent the explanation hierarchy. The device knowledge will ultimately be represented in the graphical simulation. The initial representation may be a hierarchical listing of the names of the device components or a block diagram, which can serve as a guide for constructing the sketch which will guide the construction of the graphical simulation.

8.3.7 **The Device Simulation**

The device simulation contains a graphic representation of the device structure and qualitative simulations of its functioning. Authoring the simulation starts with a temporary sketch which is derived from the prior cognitive analysis.

The construction of the simulation is done in bottom-up fashion starting with the lowest level objects in the device hierarchy. The behavior of the objects is defined by attribute handles and rules. These aspects of the simulation are drawn from the explanation hierarchy. Once the basic simulation is complete, procedures which are carried out on the device are authored by carrying out a sequence of actions which correspond to actions spelled out to accomplish the goals in the GOMS analysis. The individual actions correspond to the operators. What is missing from the simulation representation is any indication of the function or purpose, i.e. goals, of the procedure. These have to be represented in the dialogue windows.

The task analysis approach developed by Kieras and his colleagues will be used in the task analysis module of AIDA (KARS). This includes (1) a GOMS analysis for the procedural aspects of the task, (2) a mental model analysis to develop an explanation hierarchy, and (3) a decomposition of device structure and function and relating them to the GOMS analysis. Shells will aid in the analysis. A shell guides the novice in doing a GOMS analysis of a particular task using either the documentation at hand or the knowledge of a subject matter expert. Similar shells will be created for the explanation hierarchy and the device structure and function knowledge.

8.3.8 **Knowledge Base Design**

The Data Base Design Document (DBDD) describes representative frames for the AIDA knowledge bases by listing for each knowledge base the constituent frames and each frame's slots and facets.

The TRXS slot specifies the transaction shell which will be used to perform the transaction. In AIDA, there will be a TRXS for

each of the twelve primary transactions. In XAIDA, four TRXSs will be implemented. They are:

a. Identify: An identify transaction requires either an instance or class entity frame. It enables the student to acquire the names, functions, properties, and relative location of all the parts which comprise an entity. The student knows what it is.

b. Execute: An execute transaction requires either an instance or class activity frame. It enables the student to acquire the steps of an activity. The student knows how and is able to [perform] the activity.

c. Interpret: The interpret transaction requires either an instance or class process frame. It enables the student to acquire the events and causes in a process. This means that the student knows why it works and can explain the events which lead to a given consequence or can predict the consequence from a series of events.

d. Classify/Decide: A classify/decide transaction requires a superclass frame with two or more subordinate class frames each of which have two or more instance frames. These frames can be entity, activity, or process frames. It enables the student to acquire the ability to sort or classify instances as to class membership. It enables the student to know when to select an alternative.

SECTION 9. RESEARCH ISSUES

9.1 Introduction

In the Final Report for Task 0006, ALHRD listed 29 AIDA-related research issues for near- and long-term resolution. The following additional research requirements were defined during Task 0013.

9.2 Knowledge Representation

Many of the suggestions Halff made in Section 5 rest on certain assumptions concerning current maintenance practices and maintenance training. He suggested, for example, that

- equipment to be maintained can be represented within a qualitative reasoning framework;
- equipment types are decomposable into certain hierarchies defined by their structure and their functions; and
- troubleshooting practices, as taught, are derived, albeit implicitly, from fault trees.

Each of these assumptions is open to questions and none could be expected to hold for the full range of maintenance scenarios. Needed is an overall evaluation of their validity. Research to address this need would survey technical documentation and training materials for selected equipment with a view to establishing how well these assumptions are met in practice. The research would be constructive in the sense that it would produce, whenever possible, representations of the equipment that could be used in AIDA.

9.3 Skilled Troubleshooting

Studies of skilled troubleshooters are needed as another aspect of validation. The aim of these studies would be to determine whether or not the general conception of troubleshooting suggested in PTT is descriptive of skilled troubleshooters. That conception described troubleshooting as a two-stage process. In Stage 1 the troubleshooter uses a fault tree to carry the troubleshooting process either to completion or to an impasse. Stage 2, invoked only when Stage 1 reaches an impasse, employs context-free strategies like those described by Rouse or by Towne to resolve the impasse.

Research to validate this conception could be carried out by examining how skilled troubleshooters approach problems in a RAPIDS simulation. Halff sees three specific goals of these studies:

- o determine how closely experts adhere to a fault-tree approach in initial troubleshooting,
- o adopt Rouse's and/or Towne's model to account for troubleshooting faults not covered in the fault tree, and
- o determine how Stage 1 and Stage 2 troubleshooting are related (i.e., how results of Stage 1 are employed in Stage 2).

9.4 GOMS Editor

One of the most exciting developments under AIDA will be a GOMS-oriented procedure editor for RAPIDS. This editor is actually a CASE project (although authors will never know it as such) in which machine learning methods are brought to bear on example procedure traces provided by the author. Put another way, the author's tasks in specifying a procedure are to provide examples of all important cases, to designate the variables that distinguish among cases, and to choose the appropriate version of a procedure when two or more versions appear equally good. The machine's role is that of inducing or creating a GOMS model of the procedure from data provided by the author.

Development of this editor is a serious research effort in its own right. It should begin with the collection of a corpus of procedures typical of those addressed by the editor. A second stage might involve the collection of protocols from SMEs using RAPIDS in a Wizard-of-Oz mode to describe procedures in the corpus. Finally, machine-learning methods could be designed to substitute for the Wizard in order to provide a completely automatic editor.

9.5 Learning Model for GOMS

GOMS originated in studies of human-machine interfaces and is still very much a tool for interface design. Although some of Kieras' work addresses the notion of ease of learning in terms of GOMS, there is no complete GOMS-oriented account of how people learn procedures or how procedures should be taught. Since the development of GOMS representations of maintenance procedures within RAPIDS is part of the proposed AIDA, some attention should be given to developing learning and training models for procedures that are based on these representations.

9.6 Curriculum Studies

One of the most interesting issues to surface in Phase II was the position of Elaboration Theory with respect to training that addresses how-it-works knowledge. In both Sections 5.2 and 5.3 Halff recommends that students first be taught how the target equipment functions by certain exercises involving the qualitative model (e.g., asking how a change in state of one component will affect the state of another). Reigeluth pointed out at the meeting that these exercises don't epitomize anything and should, in his view, be eliminated from the curriculum. Kieras suggests that explicit how-it-works training can have positive effects on learning and performance. A series of empirical studies could serve to illuminate this issue. These studies would examine the effects of how-it-works pretraining on effectiveness of subsequent troubleshooting training.

In addition to studying the effects of how-it-works knowledge, this research would have the important side effects of providing curricula that would be directly transferable to AIDA. While Halff's papers provide a general view of the design of curricula for AIDA, they are far from specific. Many other issues, besides the how-it-works issue, will arise in the development of curricula, and the success of AIDA will depend on an integrated set of studies to address these issues.

The studies proposed above would clearly require a team effort. RAPIDS and someone with RAPIDS expertise would be needed for just about all of them. To the extent that the Air Force is interested in particular equipment, Air Force SMEs would be needed to define the equipment characteristics. Researchers versed in knowledge representation would be required for most studies. Certain individuals (Kieras and Reigeluth in particular) should be involved in some aspects of the work.

REFERENCES

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. Artificial Intelligence, 42, 7-49.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. (1984). Cognitive principles in the design of computer tutors. In Sixth Annual Conference of the Cognitive Science Society Program (pp. 2-16).

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. Science, 228, 456-462.

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. Human Computer Interaction, 5, 1-48.

Brown, J. S., Burton, R. R., & de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman & J. S. Brown (Eds.) Intelligent tutoring systems (pp. 227-282). London: Academic Press.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.) Intelligent tutoring systems (pp. 79-98). London: Academic Press.

Card, S. K., Moran, T., & Newell, A. (1983). The psychology of human computer interaction. Hillsdale, NJ: Erlbaum.

Chase, W. G., & Ericsson, K. A. (1982). Skill and Working Memory. In Bower, G. H. (Ed.), The psychology of learning and motivation (Vol. 16, pp. 1-58). New York: Academic Press.

Gagné, R. M., & Merrill, M. D. (1990) Integrative goals for instructional design. Unpublished manuscript.

Glaser, R., & Bassok, J. (1989). Learning theory and the study of instruction. In M. R. Rosenwig & L. W. Porter (Eds.), Annual Review of Psychology, (Vol. 40 pp. 631-666). Palo Alto, CA: Annual Reviews, Inc.

Halff, H. M. (1989). Prospects for automating instructional design. Arlington, VA: Halff Resources, Inc.

Halff, H. M. (1990). Automating maintenance training. Arlington, VA: Halff Resources, Inc.

Hollan, J. D., Hutchins, E. L., & Weitzman, L. (1984). STEAMER: An interactive inspectable simulation-based training system. The AI Magazine, 5(2), 15-27.

Hunt, R. M., & Rouse, W. B. (1984). A fuzzy rule-based model of human problem solving. IEEE Transactions on Systems, Man, and Cybernetics, SMC-14, 112-120.

Jones, M. K., Li, Z., & Merrill, M. D. Domain knowledge representation for instructional analysis. Educational Technology, October 1990.

Kieras, D. E. (1988b). What mental model should be taught: Choosing instructional content for complex engineered system. In J. Psotta, L. D. Massey, & S. A. Mutter (eds.), Intelligent tutoring systems: Lessons learned, pp. 85-111, Hillsdale, NJ: Erlbaum.

Kieras, D. E. (1988a). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), Handbook of Human-Computer Interaction (pp. 135-157). North Holland: Elsevier.

Kieras, D. E. (1990). The role of cognitive simulation models in the development of advanced training and testing systems. In N. Frederiksen, R. Glaser, Alan Lesgold, & M. Shafto (Eds.) Disgnostic monitoring of skill knowledge acquisition (pp. 51-74). Hillsdale, NJ: Erlbaum.

Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. Cognitive Science, 8, 255-273.

Kieras, D. E., & Polson, P. G. (1985). An approach to the formal analysis of user complexity. International Journal of Man-Machine studies 22, 365-394.

Kosslyn, S. M. (1980) Image and mind. Cambridge, MA: Harvard University Press.

Merrill, M. D., & Li, Z. An instructional design expert system. Journal of Computer-Based Instruction, Summer 1989, v. 15, no. 3, 95-101.

Polson, M. C., & Polson, P. G. (1990). AIDA: Procedural Training. Boulder, CO: Institute of Cognitive Science, University of Colorado.

Psotka, J., Massey, L. D., & Mutter, S. A. (Eds.). (1988). Intelligent tutoring systems: Lessons learned. Hillsdale, NJ: Erlbaum.

Towne, D. M. (1986). The generalized maintenance trainer: Evolution and revolution. In W. B. Rouse (Ed.), Advances in man-machine systems research (Vol 3). Greenwich, CT: JAI Press.

Towne, D. M., Johnson, M. C., & Corwin, W. H. (1983). A performance-based technique for assessing equipment maintainability (Tech. Rep. 102). Los Angeles, CA: Behavioral Technology Laboratories, University of Southern California.

Towne, D. M., & Munro, A. (1988). The intelligent maintenance training system. In J. Psotka, L. D. Massey, & S. A. Mutter (Eds.), Intelligent tutoring systems: Lessons learned. (pp. 479-530) Hillsdale, NJ: Erlbaum.

Towne, D. M., Munro, A., Pizzini, Q. A., Surmon, D. S., Coller, L. D., & Wogulis, J. L. (1990). Model-building tools for simulation-based training. Interactive Learning Environments, 1, 33-50.

U.S. Air Force. Flight Manual: USAF Series T-38A and AT-38B Aircraft. T.O. 1T-38A-1, 1 Jul 87 (Change 2, 1 May 89).

U.S. Air Force. Organizational Maintenance, Engine Conditioning: USAF Series T-38A Aircraft, T.O. 1T-38A-2-6-2, 1 Aug 77 (Change 35, 1 May 89).